Universidade Estadual de Campinas
Instituto de Computação

Gustavo Yudi Bientinezi Matsuzake

Clipped Euclidean Distance Representation of Shapes

Representação de Formas por Distância Euclidiana Truncada

CAMPINAS
2020

Gustavo Yudi Bientinezi Matsuzake

Clipped Euclidean Distance Representation of Shapes

Representação de Formas por Distância Euclidiana Truncada

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Jorge Stolfi

Este exemplar corresponde à versão final da Dissertação defendida por Gustavo Yudi Bientinezi Matsuzake e orientada pelo Prof. Dr. Jorge Stolfi.

CAMPINAS
2020

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

M429c

Matsuzake, Gustavo Yudi Bientinezi, 1994-
Clipped Euclidean distance representation of shapes / Gustavo Yudi Bientinezi Matsuzake. – Campinas, SP : [s.n.], 2020.

Orientador: Jorge Stolfi.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Transformada de distância euclidiana. 2. Modelos tridimensionais. 3. Imagens n-dimensionais. 4. Representação de formas geométricas. I. Stolfi, Jorge, 1950-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Representação de formas por distância euclidiana truncada
Palavras-chave em inglês:
Euclidean distance transform
3D models
n-Dimensional images
Representation of geometric shapes
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Jorge Stolfi [Orientador]
Sandra Eliza Fontes de Avila
Elisa de Cássia Silva Rodrigues
Data de defesa: 16-04-2020
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-0272-2619
- Currículo Lattes do autor: http://lattes.cnpq.br/2242412501193378

Universidade Estadual de Campinas
Instituto de Computação

Gustavo Yudi Bientinezi Matsuzake

Clipped Euclidean Distance Representation of Shapes

Representação de Formas por Distância Euclidiana Truncada

Banca Examinadora:

- Prof. Dr. Jorge Stolfi
  Unicamp
- Profa. Dra. Sandra Eliza Fontes de Avila
  Unicamp
- Profa. Dra. Elisa de Cássia Silva Rodrigues
  Universidade Federal de Itajubá

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 16 de abril de 2020

Acknowledgements

I thank my family; my parents Silmara and Raul and my brother Tiago for affection, care, attention, and unconditionally support.

I would like to express my gratitude to my advisor, prof. Dr. Jorge Stolfi for the exceptional tutorship, useful remarks, and patience.

This research was supported in part by the Conselho Nacional de Desenvolvimento Científico (CNPq).

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo

Nesta dissertação estudamos o uso da transformada de distância euclidiana com sinal truncada para representar a forma de objetos n-dimensionais, com ênfase em três dimensões, e para aplicações de projeto e manufatura assistidas por computador (CAD/CAM). A representação consiste de uma imagem digital onde cada pixel contém a distância de seu centro à fronteira do objeto, quantizada e truncada. Nós elaboramos ferramentas para geração dessa representação com garantias informativas sobre o interior e o exterior do objeto a ser representado, e também estudamos algoritmos para conversão de e para outras representações de formas, como imagens binárias e ternárias, polígonos e malhas de triângulos, e modelos procedurais. Por fim, investigamos os erros empíricos da extração da representação de borda através da representação de distância truncada e o impacto de seus parâmetros nessa tarefa.

Abstract

In this dissertation, we study the usage of the clipped signed distance transform to represent shapes of n-dimensional objects, with emphasis in three dimensions and for applications in computer-aided design and manufacting (CAD/CAM). The representation consists in a digital image where every pixel holds the value of its center to the boundary of the object, truncated and quantized. We elaborate tools for generating the representation with informative properties about the interior and exterior of the object being represented and examine algorithms for conversions from and to other common shape representations, like binary and ternary images, polygons, triangle meshes, and procedural models. Finally, we investigate the empirical errors of the boundary representation extraction of the clipped distance representation and the impact of its parameters for this purpose.

List of Figures

List of Tables

List of acronyms

BIR	Binary image representation
BREP	Generic boundary representation
BMR	Boundary mesh representation
CAD	Computer aided-design
CAM	Computer aided-manufacturing
CSG	Constructive solid geometry
EDR	Euclidean distance representation
PREP	Generic procedural representation
IDR	Interval distance representation
RMS	Root mean square
SDT	Signed Euclidean distance transform
TIR	Ternary image representation

Contents

Chapter 1

Introduction

The representation of the shapes of 2D and 3D objects is a critical issue in many applications, including *computer-aided design* (CAD) and *computer-aided manufacturing* (CAM). Many of those applications require applying certain geometric operations to the shapes, such as Boolean operations, gap closing, smoothing, offsetting, scaffolding, filling, slicing, and tool path planning [39, 46].

1.1 Common shape representations

Several different data structures or *shape representations* have been developed for this purpose, which have different advantages and drawbacks [2]. The choice of representation has a significant influence in the cost and convenience of processes such as CAD/CAM, and may be forced by the equipment involved. For instance, some 3D printers receive binary images as input [46] while others require lists of extrusion paths. In this section we briefly review the representations most commonly used in practice.

1.1.1 Geometric primitives

Since early days of computer graphics and CAD [42] simple shapes have been represented in the computer by a combination of *geometrical primitives*, which are simple geometric shapes such as balls, cylinders, boxes, and toruses, which in turn are defined by a small number of parameters. For instance, a spherical object can be represented by a geometric ball of $\mathbb{R}^3$, defined by the Cartesian coordinates of its center $c \in \mathbb{R}^3$ and its radius $r \in \mathbb{R}$. Similarly, an axis-aligned box can be represented by its center $c \in \mathbb{R}^3$ and its extent $s \in \mathbb{R}^3$ along the three coordinate axes.

1.1.2 Implicit function and procedural representation

Another common way to represent a shape is the *implicit function representation*, consisting of a continuous function f from $\mathbb{R}^3$ to $\mathbb{R}$ that is negative inside the shape, positive outside, and zero at the boundary. Such f is called a (*signed*) *characteristic function* for the set A.

For a ball shape, for example, one could use the function

$$f^{\mathsf{B}}(x, y, z) = (x - x')^2 + (y - y')^2 + (z - z')^2 - r^2 \tag{1.1}$$

where $c = (x', y', z')$ is the center and r is the radius of the ball. For the axis-aligned box with center $c = (x', y', z')$ and extent $s = (x'', y'', z'')$, one could use

$$f^{\mathsf{R}}(x, y, z) = \max\{|x - x'|/x'', |y - y'|/y'', |z - z'|/z''\} - 1/2. \tag{1.2}$$

A particular case of implicit function representation is the *signed Euclidean distance* d, a function that gives the distance of a point to the boundary of the object, with a negative sign if the point is inside. See Figure 1.1.

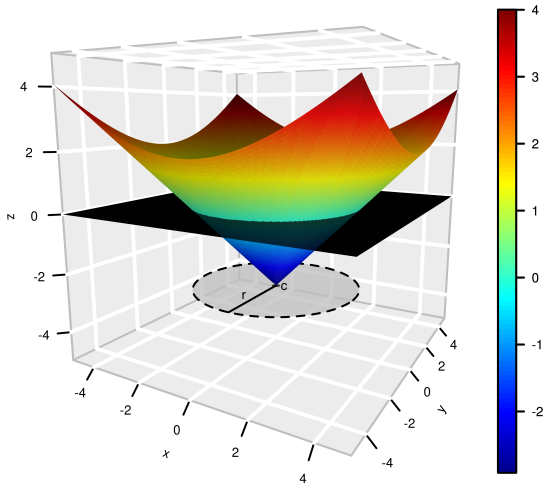


Figure 1.1: Signed distance function representation of disk centered at the origin with radius $r = 1$.

The implicit function representation for a shape is usually expressed in the computer as a *procedural representation* (PREP), a procedure that computes the value of $f(x, y, z)$. The result may be returned as a floating point number, or as an interval of values computed with interval arithmetic [34].

A major disadvantage of this representation is that implicit functions are usually difficult to construct for more complex objects.

1.1.3 Constructive solid geometry

Simpler representations can be combined with *constructive solid geometry* (CSG) operations applied to shapes. These are Boolean set operations (such as union, intersection, and difference) applied to the shapes, viewed as set of points. Thus a complex shape can be described by a tree whose internal nodes are CSG operations and whose leaves are geometric primitives, or other kinds of shape representations [6].

1.1.4 Boundary representation

Another approach to shape modeling is the *boundary representation* (BREP) which describes the boundary of the shape — rather than the interior — with a collection of simple elements of one dimension less; such as line segments and parametric arcs for 2D shapes, or triangles and Bézier patches for 3D shapes [9, 40].

1.1.5 Digital images

Yet another widely used approach is the *digital image representation*, an n-dimension array whose elements are associated with cells of a regular orthogonal grid in $\mathbb{R}^n$, with uniform size σ in each direction [4, 5, 11, 25, 26, 36]. The elements are called *pixels* for 2D images, and *voxels* for 3D ones.

In particular, in the *binary image representation* (BIR) each element of the image is 0 or 1 to indicate whether the corresponding cell is exterior or interior to the shape, respectively. See Figure 1.2.

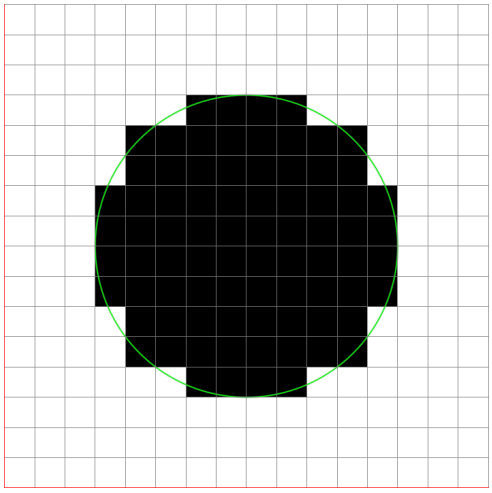


Figure 1.2: Binary image representation of a disk with radius of 5 centered at $(8, 8)$ (green outline), with cell size $\sigma = 1$. All dimensions are in mm. The image has 16×16 pixels; the grid lines show the cell boundaries.

The BIR implies that the boundary of the represented shape has "staircase" artifacts due to quantization. The object can instead be represented as an *antialiased grayscale image* where the gray level indicates the fraction of the pixels area or voxel volume that is interior to the object.

Another kind of image representation is the (*discrete*) *Euclidean distance representation* (EDR), where each pixel or voxel holds the quantized distance to the shape. This approach was introduced in 1966 and by Rosenfeld and Pfaltz, and is also called *Euclidean distance transform* (EDT) or (*discrete*) *distance field*. A variant of this approach is the *signed Euclidean distance transform* (SDT) [27], where each pixel or voxel inside the shape has the negative of the distance to the boundary of the shape.

In order to represent the shape accurately, the BIR must often have a very high *resolution*, defined as the reciprocal of the cells' size. That implies a very large memory

usage. Antialised grayscale images and the Euclidean distance representation reduce this requirement to some extent. However, the total memory size grows like $(L/\sigma)^n$ where L is the size of the object and σ is the desired accuracy.

1.2 Goals and contributions

In this dissertation, we study another variant of image representation, called the (*discrete*) *clipped signed Euclidean distance representation* (CDR), where distances to the boundary are stored only up to a certain maximum absolute value δ, usually on the order of few pixel or voxel diameters. See Figure 1.3.

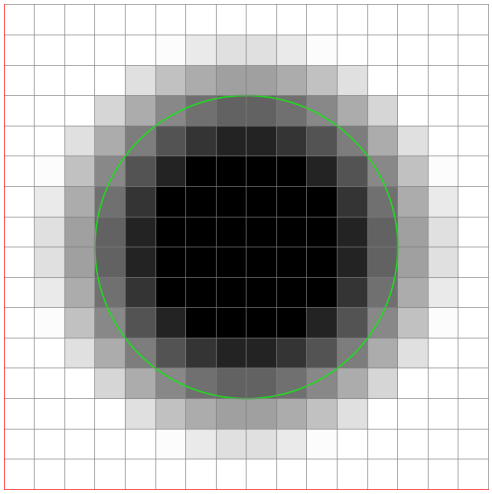


Figure 1.3: Clipped signed Euclidean distance representation of a disk with radius 5 centered at $(8, 8)$ (green outline), discretized with cell size $\sigma = 1$. All dimensions are in mm. The image has 16×16 pixels; the grid lines show the pixel boundaries.

For this representation, as for the binary image and antialiased grayscale images (but not for the exact EDR), run-length encoding arrays [24], n-dimensional binary tree (k-d *tree*) [7], or an adaptively sampled array [15] can be used to reduce the memory requirement, so that it grows like $(L/\sigma)^{n-1}$ (instead of $(L/\sigma)^n$) for most shapes that occur in practice.

In particular, we study the mathematical properties and semantics of the CDR, and algorithms for converting it to and from other representations. We also perform experimental studies of the accuracy of the CDR in comparison with the BIR.

1.3 Related work

In 1980, Requicha [40] published a survey that covers several shape representations, and listed some attributes and supported queries that a "good" representation must have. He also proposes a multi-representation approach, in which the same shape is internally represented by several methods, with the most efficient one automatically selected for each query or operation. However, the survey did not cover image-based representations, and no theoretical or empirical methods were given to compare the representations.

In 1999, Sramek and Kaufman [44] observed that, for many applications if was sufficient to compute the distance only within a certain interval $[-\delta, +\delta]$. In 2000, Bærentzen et al. discussed properties that the shape must have to be satisfactorily represented with a discrete clipped Euclidean distance image.

Algorithms have been developed for geometric operations in the discrete clipped distance representation, such as CSG operations, voxelization, sculpting, deformable modeling, and others [3, 4, 14, 35, 35, 47]. Recently, Gyulassy et al. [18], Chazal and Lieutier [12], and Regli et al. [38] discussed manufacturing issues in the context of this representation.

1.4 Document organization

We divided this dissertation in five parts,

- *Part I — Background*: is where we present then notation and fundamental concepts for this work;
- *Part II — Common shape representations*: is the definition of all common shape representations and conversions between them used in this dissertation;
- *Part III — The clipped signed distance representation*: contains the mathematical definition, properties, interpretation, and conversions of the CDR which is the main subject of this dissertation;
- *Part IV — Experiments*: is where we present the metrics and show the empirical results of the boundary extraction algorithm for BIR and CDR;
- *Part V — Conclusion*: consist of a summary of the results and discution of future work.

Part I

Background

Chapter 2

General definitions and notation

2.1 Vectors, functions, and boxes

Real subsets. We denote by $\mathbb{R}_+$ the set of all positive real numbers, and by $\mathbb{R}_{0+} = \mathbb{R}_+ \cup \{0\}$ the non-negative ones. We write $\{\}$ for the empty set, and $\overline{A}$ for the complement set $\mathbb{R}^n \setminus A$, whenever n is implied by the context.

Extended reals. We denote by $\mathbb{R}^\star$ the set $\mathbb{R} \cup \{-\infty, +\infty\}$. We extend the arithmetic operations on $\mathbb{R}$ to operations on $\mathbb{R}^\star$, in the obvious way, including $1/(+\infty) = 1/(-\infty) = 0$; except that some operations are left undefined, such as $(+\infty) + (-\infty)$, $0 \cdot (+\infty)$ $(+\infty)/(+\infty)$, and other cases reducible to the above.

Vectors. We number the axes of $\mathbb{R}^n$ from 0 to $n-1$. We denote by $\mathbf{e}_i$ the unit vector of $\mathbb{R}^n$ parallel to the axis i oriented to the positive direction of i-th axis. For example, in $\mathbb{R}^3$, $\mathbf{e}_0 = (1,0,0)$, $\mathbf{e}_1 = (0,1,0)$, and, $\mathbf{e}_2 = (0,0,1)$. The origin of $\mathbb{R}^n$ is represented by $\mathbf{0}^n$ and the all-ones vector of $\mathbb{R}^n$ is represented by $\mathbf{1}^n$. Coordinate i of a vector x will be denoted as x_i or $x[i]$.

Given two vectors u and v of $\mathbb{R}^n$, the *dot product* of u and v is denoted by $u \cdot v$, which is $\sum_{i=0}^{n-1} u_i v_i$. The *norm* of the vector v is $\sqrt{v \cdot v}$, denoted by $\|v\|$.

Closed intervals. A (*real*) *closed interval* is a subset of $\mathbb{R}$ of the form $\{\, x : a \le x \le b \,\}$ where a and b are real numbers. This set will be denoted as $[a,b]$. The interval is said to be *trivial* if $a = b$. Note that, if $a > b$, the interval $[a,b]$ is the empty set $\{\}$.

Relations. A *relation from* a set A *to* a set B is a subset of the Cartesian product $A \times B$. The *domain* of a relation R is the set $\mathcal{D}(R) = \{\, a : (\exists b)\,(a,b) \in R \,\}$, and its *range* is the set $\mathcal{R}(R) = \{\, b : (\exists a)\,(a,b) \in R \,\}$.

Functions. A *function from* a set A *to* a set B is a relation f from A to B that associates each element in A with exactly one element of the set B. Note that $\mathcal{D}(f)$ must be A, but $\mathcal{R}(f)$ may be a proper subset of B. We denote by $A \to B$ the set of all functions from A to B.

Linear functions. A function f from $\mathbb{R}^n$ to $\mathbb{R}^m$ is said to be *linear* if it satisfies $f(ax) = af(x)$ and $f(x+) = f(x)+f(y)$ for all $a \in \mathbb{R}$ and all $x, y \in \mathbb{R}^n$. Note that these conditions imply $f(-x) = -f(x)$ and $f(\mathbf{0}^n) = \mathbf{0}^m$.

Affine functions. A function f from $\mathbb{R}^n$ to $\mathbb{R}^m$ is said to be *affine* if it there is a linear function $g \in \mathbb{R}^n \to \mathbb{R}^m$ such that $f(x) = g(x) + g(\mathbf{0}^n)$ for all $x \in \mathbb{R}^n$. (These functions are sometimes improperly called "linear".)

A function $f \in \mathbb{R}^n \to \mathbb{R}^m$ is affine if and only if every coordinate of $f(x)$ can be expressed as a polynomial of degree 1 over the coordinates of x, with coefficients that do not depend on x.

Integer rounding The *ceiling* function of x denoted by $\lceil x \rceil$ rounds towards $+\infty$, i.e., returns the smallest integer z such that $z \geq x$. The *floor* of x denoted by $\lfloor x \rfloor$ is the largest integer z such that $z \leq x$.

We use the notation $\langle x \rangle$ for the value of $x \in \mathbb{R}$ rounded to the nearest integer. The rounding direction in case of ties is usually not important in practice; however, for definiteness, we will assume the round-to-even rule (-0.5 and 0.5 round to 0, 1.5 and 2.5 round to 2, 3.5 and 4.5 round to 4, etc).

Integer ranges and grids. For any natural number M we denote by $\widehat{M}$ the *integer range* $\{ z \in \mathbb{N} : 0 \leq z < M \}$, that is, $\{0, 1, \ldots, M-1\}$. If $N = (N_0, N_1, \ldots, N_{n-1})$ is a vector of n natural numbers, we denote by $\widehat{N}$ the set $\widehat{N}_0 \times \widehat{N}_1 \times \cdots \times \widehat{N}_{n-1} \subset \mathbb{N}^n$, a (*finite*) *orthogonal integer grid*.

2.2 Interpolation

Affine interpolation. The simplest way to obtain a non-trivial functional model from sampled data is *affine interpolation*. Given two argument values $x_0, x_1 \in \mathbb{R}$ and the corresponding desired values $y_0, y_1 \in \mathbb{R}$, with $x_0 \neq x_1$, their *affine interpolant* is the unique affine function $f \in \mathbb{R} \to \mathbb{R}$ such that $f(x_0) = y_0$ and $f(x_1) = y_1$; namely,

$$f(x) = y_0 + \frac{x - x_0}{x_1 - x_0}(y_1 - y_0)\,; \tag{2.1}$$

or, alternatively,

$$f(x) = y_0(1 - r) + y_1 r \tag{2.2}$$

where $r = (x - x_0)/(x_1 - x_0)$ See Figure 2.1. This function will be denoted by $f(x) = \mathrm{aff}((x_0, y_0), (x_1, y_1))(x)$.

Note that the affine interpolant is defined for any $x \in \mathbb{R}$, so it can be used to extrapolate the data outside the interval $[x_0, x_1]$. The same formula can be used for desired values y_0, y_1 in any vector space $\mathbb{V}$, such as $\mathbb{R}^m$. It also works if $x_1 < x_0$.

Multiaffine interpolation. One way to extend the concept of affine interpolation to multi-dimensional spaces is *multiaffine interpolation*. Let D be a list of given data pairs

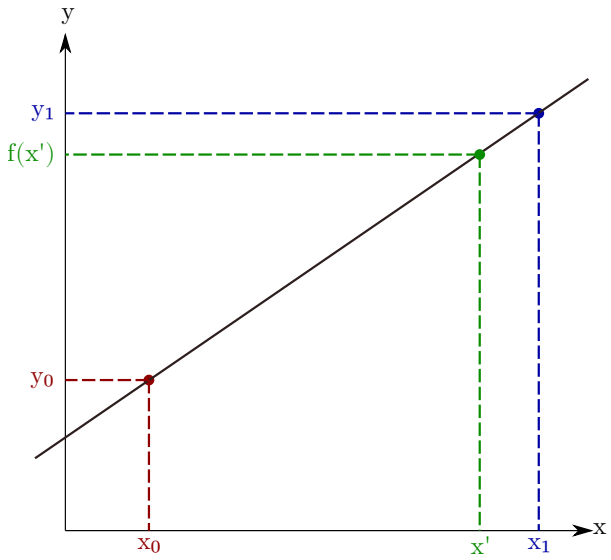


Figure 2.1: The affine interpolant f of the data pairs (x_0, y_0) and (x_1, y_1).

pairs $(p_k, z_k) \in \mathbb{R}^n \times \mathbb{R}^m$, for $k = 0, 1, \ldots, 2^n - 1$, such that the p_i are the corners of an n-dimensional box in $\mathbb{R}^n$

$$h = [a_0, b_0] \times [a_1, b_1] \times \cdots \times [a_{n-1}, b_{n-1}], \tag{2.3}$$

with $a_i < b_i$, listed in lexicographic order. The *multiaffine interpolant* of those data pairs is the function $\mathrm{maff}(D) \in \mathbb{R}^n \to \mathbb{R}^m$ defined recursively as

$$\mathrm{maff}(D)(x) = \begin{cases} z_0 & \text{if } n = 0, \\ \mathrm{aff}((a_0, \mathrm{maff}(D_0, x'), (b_0, \mathrm{maff}(D_1, x'))(x_0) & \text{if } n > 0, \end{cases} \tag{2.4}$$

where D_0 and D_1 are the first and last 2^{n-1} pairs of the list D, respectively, and x' is the vector x minus its first coordinate, that is $x' = (x_1, x_2, \ldots, x_{n-1}) \in \mathbb{R}^{n-1}$. This function can also be written as

$$\mathrm{maff}(D)(x) = \sum_{k=0}^{2^n-1} z_i \prod_{j=0}^{n-1} \begin{cases} \bar{r}_j & \text{if } p_k[j] = a_j \\ r_j & \text{if } p_k[j] = b_j \end{cases} \tag{2.5}$$

where $r_j = (x_j - a_j)/(b_j - a_j)$ and $\bar{r}_j = 1 - r_j$. These formulas apply also if $a_i > b_i$ for some or all axes i. These formulas are often but improperly called "multilinear interpolation."

Note that the multiaffine interpolant is *not* an affine function of $\mathbb{R}^n \to \mathbb{R}^m$. Each coordinate of $\mathrm{maff}(D)(x)$ is a polynomial of degree n on the coordinates of x, even though it is affine (of degree 1) on each coordinate if the others are considered fixed.

Biaffine interpolation: In particular, for $n = 2$, the data for multiaffine interpolation consists of desired values z_k at the four corners of a rectangle $[a_0, b_0] \times [a_1, b_1]$, namely

$$D = \begin{pmatrix} ((a_0, a_1), z_{00}) \\ ((a_0, b_1), z_{01}) \\ ((a_1, b_0), z_{10}) \\ ((a_1, b_1), z_{11}) \end{pmatrix}, \tag{2.6}$$

where the indices of the z values should be read in binary notation. The *biaffine interpolant* of this data is the function $\mathrm{maff}(D)$ defined by Equation (2.5), namely

$$\mathrm{maff}(D)(x) = z_{00}\bar{r}_0\bar{r}_1 + z_{01}\bar{r}_0 r_1 + z_{10} r_0 \bar{r}_1 + z_{11} r_0 r_1 \tag{2.7}$$

where, as before, $r_i = (x_i - a_i)/(b_i - a_i)$ and $\bar{r}_i = 1 - r_i$. See Figure 2.2.

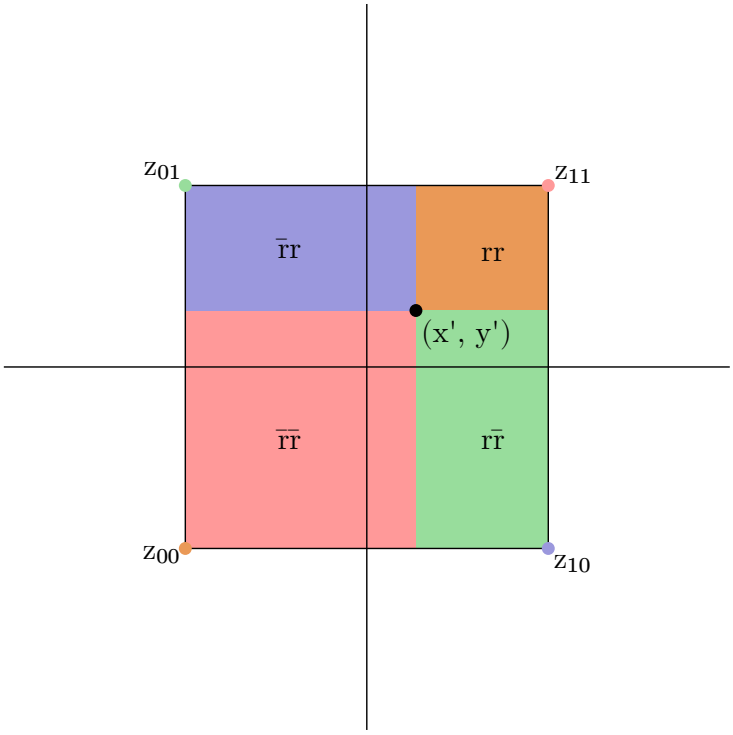


Figure 2.2: Geometric interpretation of biaffine interpolation over a rectangle R. The area of each colored sub-rectangle, relative to the area of R, is the coefficient of the given value z_k on the opposite corner of R.

Multidimensional affine interpolation: Other extension to Equations (2.1) and (2.2) from the reals to an n-dimensional domain is to look for an affine (not multiaffine) function of $\mathbb{R}^n$ that fits the given data. The latter then consists of $n+1$ arguments $p_0, p_1, \ldots, p_n \in \mathbb{R}^n$ and the corresponding desired values $z_0, z_1, \ldots, z_n$. The *affine interpolant* of the list $D = ((p_0, z_0), \ldots, (p_n, z_n))$ is then the unique affine function $\mathrm{aff}(D) \in \mathbb{R}^n \to \mathbb{R}^m$ such that

$$\mathrm{aff}(D)(x) = \sum_{k=0}^{n} z_k r_k \tag{2.8}$$

where r_0, r_1, r_n are the unique set of real numbers such that

$$\sum_{k=0}^{n} r_k = 1 \tag{2.9}$$

$$\sum_{k=0}^{n} r_k p_k = x \tag{2.10}$$

In particular, if $x = p_k$ for some k, then r_k will be 1 and all other r_j will be zero. These coefficients are called the *barycentric coordinates* of x relative to the points $p_0, p_1, \ldots, p_n$. For this reason, Equation (2.8) is also called *barycentric interpolation*.

2.3 Topology and morphology

Point set. For this dissertation, we define a *point set* as a subset of Cartesian space $\mathbb{R}^n$, where n should be specified by the context.

Neighborhood. A *neighborhood* of a point p of $\mathbb{R}^n$ is any point set X that contains an open n-dimensional ball with center p and positive radius.

Open and closed sets. We assume that the set $\mathbb{R}^n$ is a topological space with the standard topology. Namely, a point set A is said to be *open* if every point p of A has a neighborhood entirely contained in A. The set is said to be *closed* if and only if its complement is open. Note that $\{\}$ and $\mathbb{R}^n$ are the only two point sets that are both open and closed.

Interior, exterior, and boundary. The *boundary* ∂A of a point set A is the set of points of $\mathbb{R}^n$ that cannot be separated from A or from $\overline{A}$. More precisely, a point p is in ∂A if and only if every neighborhood of p has at least one point in A and one point in $\overline{A}$. The *interior* of a point set A is the set $\mathrm{int}(A)$ of all points p that have a neighborhood entirely contained in A. The *exterior* of A is the interior of the complement, namely the set $\mathrm{ext}(A)$ of all points p that have a neighborhood entirely disjoint from A.

Note that $\mathrm{int}(A)$ is contained in A and $\mathrm{ext}(A)$ is contained in $\overline{A}$, but $\partial(A)$ may be partly in A and partly in $\overline{A}$. Note also that ∂A is empty iff A is empty or $\mathbb{R}^n$. The boundary, interior, and exterior of any set A are pairwise disjoint, and their union is $\mathbb{R}^n$.

Closure The *closure* of a set A, denoted $\mathrm{cl}(A)$, is the smallest closed set that contains A. It is $A \cup \partial(A) = \mathrm{int}(A) \cup \partial(A)$.

Note that a set A is closed if and only if $A = \mathrm{cl}(A)$ and a set A is open if and only if $A = \mathrm{int}(A)$.

Bounded and co-bounded sets. A subset A of $\mathbb{R}^n$ is *bounded* if there exist a ball of $\mathbb{R}^n$ that contains it.

A is said to be *co-bounded* if $\overline{A}$ is bounded.

Closed balls. We denote with B^n the (*closed*) *unit ball*, the n-dimensional closed ball centered at the origin $\mathbf{0}^n$ with radius 1. When $n = 2$, this set will be called the (*closed*) *unit disk*.

Boxes. A (*closed*) *box of* $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that is the Cartesian product of n non-empty closed intervals. Note that, if k of those intervals are trivial, the box will be an $(n-k)$-dimensional set.

In particular, we denote by K^n the n-dimensional closed box centered at the origin $\mathbf{0}^n$ with all sides of length 1.

More generally, for any vector $h = (h_0, h_1, \ldots, h_{n-1}) \in \mathbb{R}^n_{0+}$, we write $\mathsf{K}^n(h)$ for the box with center $\mathbf{0}^n$ whose half-extent along each axis i is h_i. Namely,

$$\mathsf{K}^n(h) \;=\; [-h_0, +h_0] \times [-h_1, +h_1] \times \cdots \times [-h_{n-1}, +h_{n-1}] \tag{2.11}$$

Geometric transformations on point sets. For our purposes, a *geometric transformation* or simply *transform* is any continuous one-to-one function from $\mathbb{R}^n$ to $\mathbb{R}^n$. The *image* of a point set A by a transform T is the set $T(A) = \{\, T(a) : a \in A \,\}$.

In particular, the *translation* of a point set A by a vector $v \in \mathbb{R}^n$, denoted by $A + v$, is its image under the transform $a \mapsto a + v$, that is

$$A + v = \{\, a + v : a \in A \,\}\,. \tag{2.12}$$

Another transform is the scaling of A by some nonzero real α, denoted by αA and defined as

$$\alpha A = \{\, \alpha A : a \in A \,\} \tag{2.13}$$

Erosion and dilation. The *Minkowski sum* [20, 32] of two point sets A and B is defined as

$$A \oplus B = \bigcup_{b \in B} (A + b). \tag{2.14}$$

The operation $\oplus$ is commutative and associative, and its neutral element is the set $\{\mathbf{0}^n\}$. Analogously, the *Minkowski difference* [20] of two point sets A and B is

$$A \ominus B = \bigcap_{b \in B} (A - b). \tag{2.15}$$

We define the *dilation of* a point set A *by* a real radius $r \geq 0$, denoted by $A \oplus r$, as the Minkowski sum $A \oplus (r\mathsf{B}^n)$. Similarly, the *erosion of* a point set A *by* a scalar $r \geq 0$, denoted by $A \ominus r$, is the Minkowski difference of A and a closed ball with radius r. Note that $A \oplus 0 = A \ominus 0 = A$ for any point set. Conversely, if $A \oplus r = A$ or $A \ominus r = A$, then either $r = 0$, or A is empty, or A is full.

We extend these definitions to negative radii by the identities

$$A \oplus (-r) = A \ominus r \tag{2.16}$$

and

$$A \ominus (-r) = A \oplus r \tag{2.17}$$

for any $r > 0$.

Opening and closing. The *opening* of a point set A by a radius $r \geq 0$ is denoted by $A \circ r$ and is defined as

$$A \circ r = (A \ominus r) \oplus r. \tag{2.18}$$

The *closing* of a point set A by a radius r denoted by $A \bullet r$ is defined as

$$A \bullet B = (A \oplus r) \ominus r. \tag{2.19}$$

The point set A is *r-opened* if $A \circ r = A$ and, similarly, if A is *r-closed* if $A \bullet r = A$.

2.4 Distance concepts

Most concepts in this section could be defined for any space with any distance function, but in this dissertation we are considering only the Euclidean distance of $\mathbb{R}^n$.

Distance between points. We denote by $d(p, q)$ the Euclidean distance between two given points p, q of $\mathbb{R}^n$.

Distance from point to set. We define the distance $d(p, A)$ from a point p to a point set A as the distance from p to the closest point of A. More precisely

$$d(p, A) = \inf \{\, d(p, q) : q \in A \,\} \tag{2.20}$$

In particular, we define $d(p, A) = +\infty$ if A is empty. Note that this definition is equivalent to

$$d(p, A) = \begin{cases} 0 & \text{if } p \in A, \\ \inf \{\, d(p, q) : q \in \partial(A) \,\} & \text{if } p \notin A. \end{cases} \tag{2.21}$$

Signed distance. We define the *signed Euclidean distance* between a point p and a point set A as

$$d^+(p, A) = d(p, A) - d(p, \overline{A}) \tag{2.22}$$

$$= \begin{cases} -d(p, A) & \text{if } p \in A, \\ +d(p, A) & \text{if } p \notin A. \end{cases} \tag{2.23}$$

Note that $d^+(p, A)$ is $+\infty$ if $A = \{\}$, and $-\infty$ if $A = \mathbb{R}^n$, for all p. For any other point set $d^+(p, A)$ is a finite real number, for any p. Note also that $d^+(p, A) = 0$ if and only if $p \in \partial(A)$.

Signed distance to ball. For example, let A be the n-dimensional ball with center $c = (x', y', z') \in \mathbb{R}^3$ and radius $r \in \mathbb{R}_{0+}$; that is, $A = r\mathsf{B}^n + c$. The signed Euclidean distance from a point $p = (x, y, z) \in \mathbb{R}^3$ to A is

$$d^+(p, A) \;=\; d(p, c) - r \;=\; \sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2} - r \tag{2.24}$$

Signed distance to box. As another example, let A be the axis-aligned box with center $c = (x', y', z') \in \mathbb{R}^3$ and half-extents vector $h = (x'', y'', z'') \in \mathbb{R}^3_{0+}$; that is, $A = \mathsf{K}^3(h) + c$. The signed Euclidean distance from a point $p = (x, y, z) \in \mathbb{R}^3$ and A can be expressed as

$$d^+(p, A) = d(t, \mathbf{0}^3) - \min\{s, 0\}, \tag{2.25}$$

where

$$t = \big(\max\{0, x^*\}, \max\{0, y^*\}, \max\{0, z^*\}\big), \tag{2.26}$$

$$s = \max\{x^*, y^*, z^*\}, \tag{2.27}$$

and

$$x^* = |x - x'| - x'' \qquad y^* = |y - y'| - y'' \qquad z^* = |z - z'| - z''. \tag{2.28}$$

Clipped distance. Given any positive real number δ, we define the *δ-clipped signed distance* between a point p and a point set A as

$$d^+_\delta(p, A) = \max\big\{-\delta, \min\big\{+\delta, d^+(p, A)\big\}\big\} \tag{2.29}$$

$$= \begin{cases} \max\big\{-d(p, \overline{A}), -\delta\big\} & \text{if } p \in A, \\ \min\big\{+d(p, A), +\delta\big\} & \text{if } p \notin A. \end{cases} \tag{2.30}$$

Distance transforms. For any fixed point set $A \subseteq \mathbb{R}^n$, the distance functions from a point A can be interpreted as functions from $\mathbb{R}^n$ to $\mathbb{R} \cup \{-\infty, +\infty\}$ that are "transforms" of A.

Specifically, the *Euclidean distance transform* (EDT) of A is the map $\mathrm{D}_A \in \mathbb{R}^n \to \mathbb{R} \cup +\infty$ defined by

$$\mathrm{D}_A(p) = d(p, A) \tag{2.31}$$

The *signed Euclidean distance transform* (SDT) of A is the map $\mathrm{D}_A^+ \in \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, +\infty\}$ defined by

$$\mathrm{D}_A^+(p) = d^+(p, A) \tag{2.32}$$

Finally, for any $\delta \in \mathbb{R}_+$, the *δ-clipped signed distance transform* (CDT) of A is the function $\mathrm{D}_A^\delta \in \mathbb{R}^n \to [-\delta, +\delta]$ defined by

$$\mathrm{D}_A^\delta(p) = d_\delta^+(p, A). \tag{2.33}$$

2.5 Interval arithmetic

In this section we will briefly discuss some basic concepts of *interval arithmetic* (IA) or *interval analysis* which is a model for numerical computation invented in 1966 by Moore [33, 34]. For a more recent survey, see Hickey et al. [21]. We follow the notation and exposition in Stolfi and De Figueiredo [45].

Floating-point representation. The *floating-point representation* is an approximate encoding of real numbers that has been widely used for all sorts of scientific and engineering computations. Several representations have been used in the past, but the IEEE 754 standard for floating-point arithmetic is almost universally adopted today.

In this dissertation, "float" will mean an extended real number that can be represented exactly in the IEEE 754 double-precision floating point value, including infinities ($-\infty$, $+\infty$) but not the *not-a-number* (`NaN`) value.

We denote by $\mathbb{F} \subset \mathbb{R}$ all the finite floats, and by $\mathbb{F}^\star$ the extended set $\mathbb{F} \cup \{-\infty, +\infty\}$. Note that `NaN` is neither in $\mathbb{F}$ nor in $\mathbb{F}^\star$.

Floating point arithmetic. According to the IEEE 754 standard, a typical floating point arithmetic operation $\widetilde{\star}$ takes two numbers $x, y \in \mathbb{F}^\star$ and returns a number $z \in \mathbb{F}^\star$, which is an approximation of the exact result z of the corresponding arithmetic operation $x \star y$.

Interval. In the context of interval arithmetic, an *interval* is a pair $\widetilde{x} = [\widetilde{x}.lo, \widetilde{x}.hi]$ where $\widetilde{x}.lo, \widetilde{y}.lo \in \mathbb{F}^\star$; it represents the set $\{x \in \mathbb{R} : \widetilde{x}.lo \le x \le \widetilde{x}.hi\}$.

We denote by $\mathbb{I}$ the set of all intervals, and [] is the special interval that represent the empty set $\{\}$. In the computer, [] can be stored as any pair $[\widetilde{x}.lo, \widetilde{x}.hi]$ with $\widetilde{x}.lo > \widetilde{x}.hi$. For any $a \in \mathbb{F}$, we denote by $\widetilde{a}$ the *trivial interval* $[a, a]$ which contains only one real number — the float a itself.

Note that $[-\infty, a]$, $[a, +\infty]$, and $[-\infty, +\infty]$ are valid intervals for any $a \in \mathbb{F}$.

Interval arithmetic operations. If f is a function from $\mathbb{R}$ to $\mathbb{R}$, an *interval extension* of f is a function $\widetilde{f}$ from $\mathbb{I}$ to $\mathbb{I}$ such that, for all $\widetilde{x} \in \mathbb{I}$,

$$\{f(x) : x \in \widetilde{x} \cap \mathcal{D}(f)\} \subseteq \widetilde{f}(\widetilde{x}). \tag{2.34}$$

Similarly, if $\star$ is an operator $\mathbb{R} \times \mathbb{R}$ to $\mathbb{R}$ such as $\{+, -, \times, \div\}$, an interval extension of this operator $\widetilde{\star}$ is an operator from $\mathbb{I} \times \mathbb{I}$ to $\mathbb{I}$, such that, given two intervals $\widetilde{x}, \widetilde{y}$ and the arithmetic operation $\star \in \{+, -, \times, /\}$, the result of the operation on the intervals $\widetilde{x}, \widetilde{y}$ is an interval $\widetilde{x} \widetilde{\star} \widetilde{y}$ such that

$$\{x \star y : x \in \widetilde{x}, y \in \widetilde{y}\} \subseteq \widetilde{x} \widetilde{\star} \widetilde{y}. \tag{2.35}$$

Ideally, $\widetilde{f}$ and $\widetilde{x} \widetilde{\star} \widetilde{y}$ should be the smallest intervals that satisfy Equations (2.34) and (2.35), but in practice that interval may be too expensive to compute, therefore a somewhat wider interval is often obtained instead.

A properly coded interval arithmetic extension $\widetilde{f}(\widetilde{x}, \widetilde{y}, \dots)$ of a real-valued function $f(x, y, \dots)$ should be *information-monotonic*, meaning that

$$\widetilde{f}(\widetilde{x}', \widetilde{y}', \dots) \subseteq \widetilde{f}(\widetilde{x}'', \widetilde{y}'', \dots) \tag{2.36}$$

if $\widetilde{x}' \subseteq \widetilde{x}''$, $\widetilde{y}' \subseteq \widetilde{y}''$, etc. are all true. That is, computing the function with less accurate arguments should not yield a more accurate result.

In order to simplify the notation, in the remainder of this dissertation we will denote IA operations with the same symbols of the corresponding real operations; namely, $+$, $/$, $\sqrt{\ }$, etc. instead of $\widetilde{+}$, $\widetilde{/}$, $\widetilde{\sqrt{\ }}$, etc. Also, if a is any value in $\mathbb{F}$, we will write a instead of $\widetilde{a} = [a, a]$ in IA computations.

2.6 Useful interval arithmetic procedures

Algorithms for the basic arithmetic operations were given byMoore [33] and reviewed by Stolfi and De Figueiredo [45]. In this section we give algorithms for some additional operations needed later on.

Directed rounding. A feature provided by IEEE 754 standard is the rounding mode control [23] which lets the programmer choose the rounding direction of floating-point operations. This feature is very helpful when implementing reliable arithmetic operations for intervals [45].

Given an expression $\mathcal{E}$ we will denote as $\langle \mathcal{E} \rangle$ the result of computing $\mathcal{E}$ in floating point with the round-to-nearest rule (or to even in case of ties). We also denote as $\uparrow \mathcal{E} \uparrow$ the result obtained by computing $\mathcal{E}$ with upward rounding and similarly as $\downarrow \mathcal{E} \downarrow$ the result with downward rounding. In order to avoid ambiguity, we will usually limit $\mathcal{E}$ to a single arithmetic operation.

2.6.1 Absolute value

Algorithm 1 is a straightforward IA implementation of the absolute value function $|x|$. The floating-point absolute value operations `abs` in steps Lines 2 and 3 are safe because the IEEE standard states that sign negation is exact, for any float value, and never overflows [23, 45].

Algorithm 1: `IA.ABS`

Input: An interval $\widetilde{x}$
Output: An interval that has the absolute value of every real $x \in \widetilde{x}$

```
if x̃ = [] then
    return []
lo = min(abs(x̃.lo), abs(x̃.hi)))
hi = max(abs(x̃.lo), abs(x̃.hi)))
if x̃.lo < 0 and x̃.hi > 0 then
    return [0, hi]
else
    return [lo, hi]
```

2.6.2 Maximum and minimum

Algorithm 2 and Algorithm 3 compute the maximum and minimum between two intervals, respectively.

Algorithm 2: `IA.MAX`

Input: Two intervals $\widetilde{a}$ and $\widetilde{b}$
Output: An interval that has $\max\{x, y\}$ for all $x \in \widetilde{a}$ and $y \in \widetilde{b}$.

```
if ã = [] or b̃ = [] then
    return []
return [max(ã.lo, b̃.lo), max(ã.hi, b̃.hi)]
```

Algorithm 3: `IA.MIN`

Input: Two intervals $\widetilde{a}$ and $\widetilde{b}$
Output: An interval that has $\min\{x, y\}$ for every $x \in \widetilde{a}$ and $y \in \widetilde{b}$.

```
if ã = [] or b̃ = [] then
    return []
return [min(ã.lo, b̃.lo), min(ã.hi, b̃.hi)]
```

2.6.3 Euclidean distance

Algorithm 4 computes the interval extension of the Euclidean distance between two points whose coordinates are intervals. The function `IA.SQR` computes the square of the argument with IA [45].

2.7 Image concepts

Image array. We define an *n-dimensional image* as a function `I` from some finite integer grid $\widehat{N} \subset \mathbb{Z}^n$ to some finite set V of values, for some $N = (N_0, N_1, \ldots, N_{n-1}) \in \mathbb{N}^n$.

Algorithm 4: `IA.EUCLIDEAN-DISTANCE`

Input: Two points $\widetilde{p}, \widetilde{q} \in \mathbb{I}^n$
Output: An interval that has $d(p,q)$ for every $p \in \widetilde{p}$ and $q \in \widetilde{q}$

```
1 d̃ = [0, 0]
2 for i in {0, ..., n − 1} do
3 |   t̃ = p̃[i] − q̃[i]
4 |   d̃ = d̃ + IA.SQR(t̃)
5 return √d̃
```

The vector N is the *size* and the orthogonal grid $\widehat{N}$ is its *domain*, denoted $\mathcal{D}(\mathtt{I})$. We use the notation $\mathtt{I}[p]$ instead of $\mathtt{I}(p)$ for the value of the function at a point $p \in \widehat{N}$.

Typically, the value set V is a finite range of integers, or a floating-point number space such as $\mathbb{F}^\star$. In particular, a *binary image* `I` is an image whose values are 0 or 1.

Storage size. The total storage size of such an image array, with the straightforward array data structure, is approximately $b \prod_i N_i$ bits, where b is the number of bits needed to represent any value in the set V. If the element values are constant over large regions of the domain, the storage size can be considerably reduced by using a more sophisticated data structure, such as a quadtree (for $n = 2$) or an octree (for $n = 3$) [30].

Pixel and voxel. We define as an *element of an image* `I`, the tuple $e = (p, v)$ where p is in the domain of `I` and v is the value of `I` at p. We denote as *pixels* the elements of a 2D image, and *voxels* the elements of a 3D image. For convenience, we also may use voxel to elements of an n-dimensional image.

Part II

Common shape representations

Chapter 3

Common shape representations

In this chapter, we will define common shape representations considered in this dissertation, and related concepts.

3.1 Interval distance representation (IDR)

In general, a *procedural representation* of a shape $A \subseteq \mathbb{R}^n$ is a procedure that tells whether a point of $\mathbb{R}^n$ is inside or outside A, by computing some signed characteristic function of A.

For this dissertation, we consider a specific form of procedural representation, namely the *interval distance representation* (IDR). It consists of an algorithm $\widetilde{d}$ that, given any n-dimensional interval $\widetilde{p} \in \mathbb{I}^n$, returns an interval $\widetilde{d}(\widetilde{p}) \in \mathbb{I}$ that contains $d(p)$ for any $p \in \widetilde{p}$; where d is either the signed Euclidean distance $d^+(p, A)$ or a δ-clipped version $d^+_\delta(p, A)$. The algorithm $\widetilde{d}$ need not be exact; it need only be sufficiently precise — that is, the returned intervals are narrow enough — for the intended application.

If the interval $\widetilde{x} = \widetilde{d}(\widetilde{p})$ has the upper bound $r = \widetilde{x}.hi < 0$, then every point $p \in \widetilde{p} \subseteq \mathbb{R}^n$ is in the interior of A, and its distance from the boundary ∂A is at least $-r$. For this reason, we can safely assume that the Minkowski sum $(-r)\mathsf{B}^n + \widetilde{p}$ is inside the representation. Similarly, if $s = \widetilde{x}.lo > 0$, the entire box $\widetilde{p}$ is contained in the exterior to A, and its distance from the boundary is at least s.

Accordingly, we can define the *interior* and *exterior* of the shape *determined by* the procedure $\widetilde{d}$ as

$$\operatorname{int}(\widetilde{d}) = \bigcup_{\substack{p \in \mathbb{F}^n \\ \widetilde{d}(p).hi < 0}} \operatorname{int}\left((-\widetilde{d}(p).hi)\mathsf{B}^n + p \right), \tag{3.1}$$

and

$$\operatorname{ext}(\widetilde{d}) = \bigcup_{\substack{p \in \mathbb{F}^n \\ \widetilde{d}(p).lo > 0}} \operatorname{int}\left((\widetilde{d}(p).lo)\mathsf{B}^n + p \right). \tag{3.2}$$

These definitions are conservative, in the sense that $\operatorname{int}(\widetilde{d}) \subseteq \operatorname{int}(A)$ and $\operatorname{ext}(\widetilde{d}) \subseteq \operatorname{ext}(A)$. If the procedure is correct, then $\operatorname{int}(\widetilde{d}) \cap \operatorname{ext}(\widetilde{d}) = \{\}$. We also define

$$\partial(\widetilde{d}) = \mathbb{R}^n \setminus \big(\operatorname{int}(\widetilde{d}) \cup \operatorname{ext}(\widetilde{d})\big). \tag{3.3}$$

If $\widetilde{d}(\widetilde{p}).lo = 0$, the box $\widetilde{p}$ *may* contain points of ∂A, but will not contain any interior point. Similarly, if $\widetilde{d}(\widetilde{p}).hi = 0$, the box $\widetilde{p}$ may contain points of ∂A, but not of the exterior. If $\widetilde{d}(\widetilde{p}) = [0, 0]$, then the entire box $\widetilde{p}$ is contained in ∂A; this is possible only if $\widetilde{p}$ itself has empty interior.

On the other hand, if $\widetilde{d}(\widetilde{p}).lo < 0 < \widetilde{d}(\widetilde{p}).hi$, the situation of $\widetilde{p}$ relative to the object is uncertain: from that result, we cannot tell whether $\widetilde{p}$ it contains any points of the interior, of the exterior, or of the boundary of A.

In general, the procedure $\widetilde{d}$ gives only a rough approximation of the shape A, because the space between $\operatorname{int}(\widetilde{d})$ and $\operatorname{ext}(\widetilde{d})$ can be bigger than zero, therefore we do not have enough information to specify exactly where the boundary of $\widetilde{d}$ is. However, we can tell that $\partial A \subseteq \partial(\widetilde{d})$.

While the set $\mathbb{F}^n$ is finite, it is too large to be enumerated in reasonable time. Therefore the definitions Equations (3.1) and (3.2) are only theoretical concepts. In practice, $\widetilde{d}$ will be evaluated for a small enough set of boxes $\widetilde{p} \in \mathbb{I}^n$ that are known to cover the set A or its complement. The "interior" and "exterior" of A that can be deduced from such probes will usually be proper subsets of $\operatorname{int}(\widetilde{d})$ and $\operatorname{ext}(\widetilde{d})$.

3.1.1 Example: ball

Algorithm 5 is an IDR of the object $r\mathsf{B}^n + c$, where the coordinates of the center $c \in \mathbb{R}^n$ and the radius $r \in \mathbb{R}$ may be given by floating point numbers or as intervals. It uses Algorithm 4, `IA.EUCLIDEAN-DISTANCE` to compute the signed Euclidean distance from $\widetilde{p} \in \mathbb{I}^n$ to the ball.

Algorithm 5: `PROCEDURAL-BALL`

Input: An interval point $p \in \mathbb{I}^n$, and shape parameters $\widetilde{c} \in \mathbb{I}^n$ and $\widetilde{r} \in \mathbb{I}$.

Output: An interval $\widetilde{x}$ that contains $d^+(p, r\mathsf{B}^n + c)$ for any $p \in \widetilde{p}$, any $c \in \widetilde{c}$, and $r \in \widetilde{r}$.

1 **return** `IA.EUCLIDEAN-DISTANCE`$(\widetilde{p}, \widetilde{c}) - \widetilde{r}$

3.1.2 Example: box

Algorithm 6 is an IDR of the axis-aligned box $A = \mathsf{K}^n(h) + c$, where c is the center of the box and $h = (h_0, \ldots, h_{n-1}) \in \mathbb{R}^n_{0+}$ is a vector with the half-sides of the box along the n coordinate axes. As in the ball example, the parameters c and r may be given as floating-point values or as intervals.

The procedure is a IA implementation of the mathematical formula for signed Euclidean distance $d^+(p, A)$, as given in Equations (2.25) – (2.28). It uses temporary variables $\widetilde{a}, \widetilde{d} \in \mathbb{I}$ and $\widetilde{t} \in \mathbb{I}^n$.

3.1.3 Example: box with rounded edges

A more interesting example is an axis-aligned box whose edges and corners have been rounded off with cylindrical and spherical surfaces, respectively, of the same radius r.

Algorithm 6: PROCEDURAL-BOX

Input: An interval point $\widetilde{p} \in \mathbb{I}^n$, and the shape parameters $\widetilde{c}, \widetilde{h} \in \mathbb{I}^n$
Output: An interval that contains $d^+(p, \mathsf{K}^n(h) + c)$ for any $p \in \widetilde{p}$, any $c \in \widetilde{c}$, and any $h \in \widetilde{r}$.

1 **for** i **in** $\{0, \ldots, n-1\}$ **do**
2 $\quad \widetilde{a} = \texttt{IA.ABS}(\widetilde{p}[i] - \widetilde{c}[i]) - \widetilde{h}[i]$
3 $\quad$ **if** $i = 0$ **then** $\widetilde{d} = \widetilde{a}$ **else** $\widetilde{d} = \texttt{IA.MAX}(\widetilde{d}, \widetilde{a})$
4 $\quad \widetilde{t}[i] = \texttt{IA.MAX}(\widetilde{a}, 0)$
5 **return** $\texttt{IA.EUCLIDEAN-DISTANCE}(\mathbf{0}^n, \widetilde{t}) + \texttt{IA.MIN}(\widetilde{d}, 0)$

Mathematically, this shape can be written $A = \big(\mathsf{K}^n(h) \oplus r\big) + c$, where $c \in \mathbb{R}^n$ is the center of the box, and $h \in \mathbb{R}^n_{0+}$ is such that $h_i + r$ is its half-extent along each axis i. The IDR is based on the following result[5]:

Theorem 1. *If A is convex subset of $\mathbb{R}^n$, then, for any $r \in \mathbb{R}_{0+}$, $d^+(p, A \oplus r) = d^+(p, a) - r$.*

The proof of this theorem is based on the fact that, if q is the point of ∂A nearest to a point p (interior or exterior), the point q' of $\partial(A \oplus r)$ nearest to p lies on the line pq, at distance r from q. The IA implementation of $d^+(p, A \oplus r)$ is Algorithm 7.

Algorithm 7: PROCEDURAL-ROUNDED-BOX

Input: A interval point $p \in \mathbb{I}^n$ and the shape parameters $\widetilde{c}, \widetilde{h} \in \mathbb{I}^n$, $\widetilde{r} \in \mathbb{I}$.
Output: An interval that contains $d^+\big(p, \mathsf{K}^n(h) \oplus r) + c\big)$, for any $p \in \widetilde{p}$, any $h \in \widetilde{h}$, any $r \in \widetilde{r}$, and any $c \in \widetilde{c}$.

1 **return** $\texttt{PROCEDURAL-BOX}(\widetilde{p}, \widetilde{h}, \widetilde{c}) - \widetilde{r}$

3.2 Digital image representation

Image arrays are commonly used for representation of n-dimensional volume data. This is usually done by associating every value of the image array domain to some spatial information.

For this purpose, we are defining a (*digital*, *discrete*) *image representation* $\mathbf{I}$ as a tuple $(\mathtt{I}, \sigma, v, \dots)$, where $\mathtt{I}$ is an image array of size N and values in V, σ is a positive real (the *cell size*), and v is an element of V (the *surround value*). For any $p \in \mathbb{Z}^n$, we define

$$\mathbf{I}[p] = \begin{cases} \mathtt{I}[p] & \text{if } p \in \mathcal{D}(\mathtt{I}), \\ v & \text{if } p \notin \mathcal{D}(\mathtt{I}). \end{cases} \tag{3.4}$$

Each element $\mathbf{I}[p]$ is related to a *cell* of an infinite n-dimensional orthogonal grid in $\mathbb{R}^n$ with step σ; specifically, to the hypercube with side σ centered at the point

$$\mathbf{I}^{\boxdot}[p] = \left(p + \frac{1}{2}\mathbf{1}^n\right)\sigma, \tag{3.5}$$

that is

$$\mathbf{I}^{\square}[p] = \sigma\mathsf{K}^n + \mathbf{I}^{\boxdot}[p]. \tag{3.6}$$

Thus $\mathbf{I}^{\square}[p]$ is the (closed) axis-aligned hypercube of $\mathbb{R}^n$ with corners at σp and $\sigma(p+\mathbf{1}^n)$, that is, $[\sigma p_0, \sigma(p_0+1)] \times [\sigma p_1, \sigma(p_1+1)] \times \cdots \times [\sigma p_{n-1}, \sigma(p_{n-1}+1)]$.

The *domain box* $\mathcal{B}(\mathbf{I})$ of the image representation is the axis aligned box of $\mathbb{R}^n$ that is the union of all cells $\mathbf{I}^{\square}[p]$, for all p in the domain $\mathcal{D}(\mathtt{I})$ of the image array. In other words, $\mathcal{B}(\mathbf{I})$ is the set

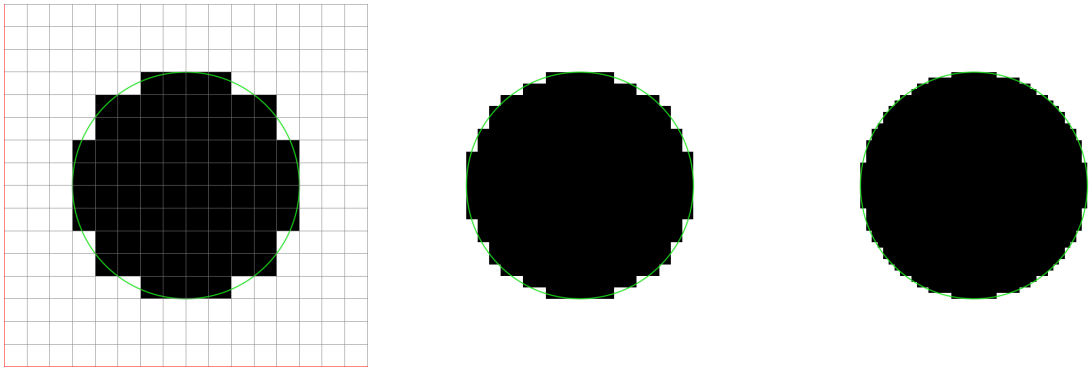
$$\mathcal{B}(\mathbf{I}) = [0, \sigma N_0] \times [0, \sigma N_1] \times \cdot\cdot\, [0, \sigma N_{n-1}]. \tag{3.7}$$

In many CAD/CAM and other applications, the cell size σ is expressed in terms of some unit of length, such as millimeters; and *resolution* of the image representation, being the reciprocal of σ, is then expressed as cells per unit of length, in that case cell/mm or just mm^{-1}.

3.3 Binary image representation (BIR)

A *binary image representation* (BIR) is an image representation whose element value set is $V = \{0, 1\}$. By definition, the *interior*, *exterior*, and *boundary* of a BIR $\mathbf{B} = (\mathtt{B}, \sigma, v)$ are the sets

$$\mathrm{int}(\mathbf{B}) = \mathrm{int}\left(\bigcup_{\substack{p \in \mathbb{Z}^n \\ \mathbf{B}[p]=1}} \mathbf{B}^{\square}[p]\right), \tag{3.8}$$



(a) $1/\sigma = 1\,\mathrm{mm}^{-1}$ (b) $1/\sigma = 2\,\mathrm{mm}^{-1}$ (c) $1/\sigma = 4\,\mathrm{mm}^{-1}$

Figure 3.1: Binary image representations of a disk with radius 5 centered at $(8, 8)$ (green outline), with various cell sizes σ. All dimensions are in mm. Each image has $N \times N$ pixels, with $N = 16/\sigma$. The grid lines show the pixel boundaries.

$$\mathrm{ext}(\mathbf{B}) = \mathrm{int}\left(\bigcup_{\substack{p\in\mathbb{Z}^n \\ \mathbf{B}[p]=0}} \mathbf{B}^{\square}[p]\right), \tag{3.9}$$

and

$$\partial(\mathbf{B}) = \mathbb{R}^n \setminus (\mathrm{int}(\mathbf{B}) \cup \mathrm{ext}(\mathbf{B}))\,. \tag{3.10}$$

The boundary of $\mathbf{B}$ is the union of all faces of cells that lie adjacent to at least one cell with value 0 and at least one cell with value 1.

This representation can be used to approximate an object A by assigning $\mathtt{B}[p] = 1$ if $\mathrm{int}(\mathbf{B}^{\square}[p]) \subseteq \mathrm{int}(A)$, $\mathtt{B}[p] = 0$ if $\mathrm{int}(\mathbf{B}^{\square}[p]) \subseteq \mathrm{ext}(A)$, and an arbitrary value otherwise (when $\mathrm{int}(\mathbf{B}^{\square}[p])$) intersects both $\mathrm{int}(A)$ and $\mathrm{ext}(A)$). See Figure 3.1.

The inherent error of this representation comes from the fact that pixels that straddle the boundary are stored as 0 or 1, which causes $\mathrm{int}(\mathbf{B})$ to be different from $\mathrm{int}(A)$.

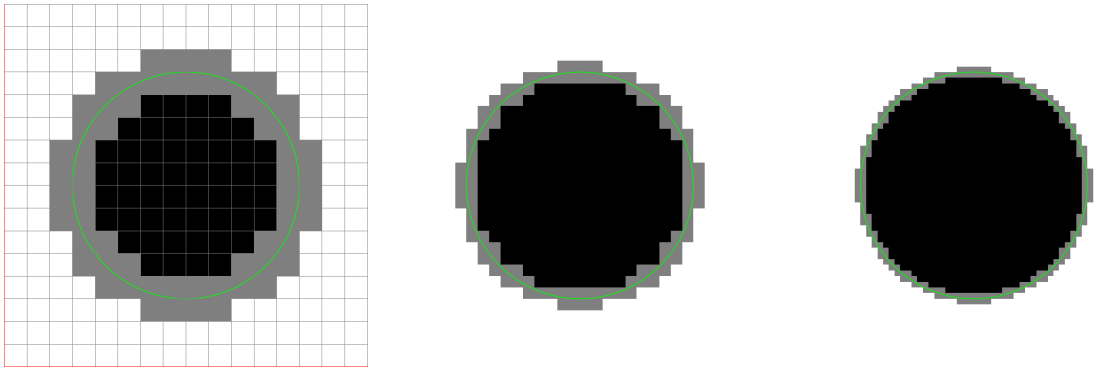
As noted before, the storage size of this representation is approximately $\prod_i N_i$ bits. If the object has a simple shape, by using quadtrees or octrees the storage size can be reduced to $O(\prod_i N_i^{(n-1)/n})$ bits, possibly times a logarithmic factor.

3.4 Ternary image representation (TIR)

The binary image representation forces one to assign every cell to the interior or exterior, even cells that intersect both or touch the boundary, or whose situation is unknown. In other words, by using the BIR one is forced to lie about the intended shape.

The unknown or mixed state of such cells can be preserved by using a *ternary image representation* (TIR), an image representation with value set $V = \{-1, 0, +1\}$.

The idea is that each element of the TIR $\mathbf{T} = (\mathtt{T}, \sigma, v)$ says whether the cell $\mathbf{T}^{\square}[p]$ is known to be completely inside a target shape A (-1), is known to be completely outside A $(+1)$, or it either intersects the boundary or its state is unknown (0). Thus, we define



(a) $1/\sigma = 1\,\mathrm{mm}^{-1}$ (b) $1/\sigma = 2\,\mathrm{mm}^{-1}$ (c) $1/\sigma = 4\,\mathrm{mm}^{-1}$

Figure 3.2: Ternary image representations of a disk with radius 5 centered at $(8, 8)$, with various cell sizes σ. All dimensions are in mm. Each image has size $N \times N$ where $N = 16/\sigma$. The grid lines show the pixel boundaries.

the *interior*, *exterior*, and *boundary* of $\mathbf{T}$ as

$$\mathrm{int}(\mathbf{T}) = \bigcup_{\substack{p \in \mathbb{Z}^n \\ \mathbf{T}[p] = -1}} \mathbf{T}^{\square}[p], \tag{3.11}$$

$$\mathrm{ext}(\mathbf{T}) = \bigcup_{\substack{p \in \mathbb{Z}^n \\ \mathbf{T}[p] = +1}} \mathbf{T}^{\square}[p], \tag{3.12}$$

and

$$\partial\mathbf{T} = \mathbb{R}^n \setminus (\mathrm{int}\mathbf{T} \cup \mathrm{ext}\mathbf{T}) \tag{3.13}$$

$$= \mathrm{int}\left(\bigcup_{\substack{p \in \mathbb{Z}^n \\ \mathbf{T}[p] = 0}} \mathbf{T}^{\square}[p] \right) \tag{3.14}$$

By construction, no cell face is adjacent to cells with values $+1$ and -1.

Therefore given a proper TIR $\mathbf{T}$ of a shape A, we can say that

$$\mathrm{int}(\mathbf{T}) \subseteq \mathrm{int}(A), \tag{3.15}$$

$$\mathrm{ext}(\mathbf{T}) \subseteq \mathrm{ext}(A), \tag{3.16}$$

and

$$\partial A \subseteq \partial\mathbf{T}. \tag{3.17}$$

See Figure 3.2.

3.5 Boundary mesh representation (BMR)

In general, a *boundary representation* (BREP) of a shape A is an explicit formula or data structure that can be used to generate the points ∂A. This representation is practical for CAM/CAD only if ∂A is a manifold of dimension $n-1$ in $\mathbb{R}^n$: namely, a curve when $n = 2$ and a surface when $n = 3$.

For example, if A is the closed square of $\mathbb{R}^2$ with corners a, b, c, d, the representation can be the set of the four line segments $\{ab, bc, cd, da\}$, each defined by four coordinates. If A is the unit ball of $\mathbb{R}^3$, a BREP can be the function f from $[0, +\pi) \times [-\pi/2, +\pi/2]$ to $\mathbb{R}^3$ with formula $f(\varphi, \lambda) = (\cos\varphi \cos\lambda, \cos\varphi \sin\lambda, \sin\varphi)$. By varying the parameters φ and λ, the point $f(\varphi, \lambda)$ will sweep the unit sphere.

In this dissertation, we consider only one type of BREP, namely a *boundary simplicial mesh representation* (BMR). It is a pair $\mathbf{T} = (\mathtt{T}, v)$, where $\mathtt{T}$ is a set of (closed) $(n-1)$-dimensional simplices in $\mathbb{R}^n$, and v is either $+1$ or -1. Each simplex is defined by the coordinates of its n vertices. It is required that the intersection of any two simplices be either empty, or a single shared face of dimension at most $n-2$.

We define the *boundary* of the mesh $\mathbf{T}$ as

$$\partial\mathbf{T} = \bigcup \mathtt{T} \tag{3.18}$$

This set is required to be a bounded, borderless, and orientable $(n-1)$-manifold in $\mathbb{R}^n$. It therefore divides the space $\mathbb{R}^n$ in two or more regions, with exactly one them being unbounded. If v is $+1$, the *exterior* $\mathrm{ext}(\mathbf{T})$ of the mesh consists of the unbounded region, and of all points in $\mathbb{R}^n \setminus \partial A$ that can be reached from a point of it by a path that crosses an even number of simplices of $\mathtt{T}$, and each of them at a single point in a different moment. The remaining points of $\mathbb{R}^n \setminus \partial A$ are, by definition, the *interior* $\mathrm{int}(\mathbf{T})$ of the mesh. If v is -1, the two sets are swapped. Thus, the shape $\mathrm{int}(\mathbf{T}) \cup \partial\mathbf{T}$ represented by $\mathbf{T}$ is bounded if $v = +1$, and co-bounded if $v = -1$.

The *elements* of the mesh are it simplices, and all their faces of dimensions 0 through $n-2$, Elements of dimension 0, 1, 2, and 3 (if they exist) are called *vertices*, *edges*, *faces*, and *cells* of the mesh, denoted $\mathcal{V}\mathbf{T}$, $\mathcal{E}\mathbf{T}$, $\mathcal{F}\mathbf{T}$, $\mathcal{C}\mathbf{T}$, respectively. The requirements on $\partial\mathbf{T}$ imply certain constraints on the way these elements may be shared by other elements.

In some applications, additional restrictions are placed on the mesh. For instance, there may be constraints on the order in which the vertices of each simplex are listed; or the mesh must include explicit information about which faces share edges or vertices. Constructing and using such information is outside the scope of this dissertation.

Chapter 4

Conversion from IDR to simple images

In this chapter, we will describe some conversions algorithms between a procedural representation of a shape — specifically, an interval version of the signed Euclidean distance transform (IDR) — to the binary and ternary image representations. These algorithms are useful as intermediate steps in conversion between other formats.

4.1 Conversion to BIR

In this section we show a method to convert an IDR $\widetilde{h}$ of a bounded or co-bounded object A to a BIR $\mathbf{I}$ for the same object. The method uses simple *point sampling* at cell centers.

The algorithm requires an image size N that is large enough to contain the boundary of A plus several cell sizes in every direction. The value of the image at $\mathbf{I}[p]$ is to 1 if $\mathbf{I}^{\boxdot}[p]$ is inside the object, according to $\widetilde{h}$, or set to 0 otherwise. See Algorithm 8.

Algorithm 8: `PROCEDURAL-TO-BIN`

Input: An IDR $\widetilde{h} \in \mathbb{I}^n \to \mathbb{I}$ of a target object A; a positive cell size $\sigma \in \mathbb{F}$; a domain size $N \in \mathbb{N}^n$; and a surround value $v \in \{0,1\}$.

Output: A BIR $\mathbf{I}$ for A, with those parameters.

```
I = NEW-IMG-ARRAY(N, {0,1})
for each p ∈ N̂ do
    c = σ (p + ½ 1^n)
    x̃ = h̃(c)
    if x̃.hi ≤ 0 then I[p] = 1 else I[p] = 0
return (I, σ, v)
```

Note that the cell $\mathbf{I}^{\square}[p]$ of the BIR may not be completely inside $\mathrm{int}(\widetilde{h})$, in other words, there are cases where $\mathrm{int}(\mathbf{I}) \not\subseteq \mathrm{int}(\widetilde{h})$. This occurs for the reason that we are only considering the center of the cell.

To alleviate this problem, one could evaluate $\widetilde{x} = \widetilde{h}(\widetilde{c})$ where $\widetilde{c} = \mathbf{I}^{\square}[p] \in \mathbb{I}^n$, and set $\mathbf{I}[p]$ to 1 if $\widetilde{\widetilde{x}}.hi \leq 0$, to 0 if $\widetilde{\widetilde{x}}.lo \geq 0$, and or to an arbitrary value otherwise. However, it will still be impossible to ensure both $\mathrm{int}(\mathbf{I}) \subseteq \mathrm{int}(\widetilde{h})$ and $\mathrm{ext}(\mathbf{I}) \subseteq \mathrm{ext}(\widetilde{h})$.

4.2 Conversion to TIR

In this section, we present an algorithm to convert an IDR $\widetilde{h}$ of an n-dimensional object A to a TIR $\mathbf{T}$ in such a way to ensure that $\mathrm{int}(\mathbf{T}) \subseteq \mathrm{int}(\widetilde{h})$ and $\mathrm{ext}(\mathbf{T}) \subseteq \mathrm{ext}(\widetilde{h})$.

The procedure is based on the idea described at the end of the previous section. Namely, evaluate $\widetilde{x} = \widetilde{h}(\widetilde{c})$ where $\widetilde{c} = \mathbf{I}^{\square}[p] \in \mathbb{I}^n$, and set $\mathbf{I}[p]$ to -1 if $\widetilde{\widetilde{x}}.hi < 0$, and to $+1$ if $\widetilde{\widetilde{x}}.lo > 0$; but, if neither condition is true, we set $\mathbf{I}[p]$ to zero (instead of an arbitrary value). This approach is described in Algorithm 9.

We test whether a cell $\mathbf{T}^{\square}[p]$ is completely inside the object comparing the distance $\widetilde{x} = \widetilde{h}(\mathbf{T}^{\boxdot}[p])$ with the diagonal h. If the absolute value of $\widetilde{x}$ is greater or equal to h then the cell is completely inside (see Line 5) or outside (see Line 7). Otherwise, the relation to the boundary is unknown (see Line 10).

Algorithm 9: `PROCEDURAL-TO-TIR`

Input: An IDR $\widetilde{h} \in \mathbb{I}^n \to \mathbb{I}$ of a target object A; a positive cell size $\sigma \in \mathbb{F}$; a domain size $N \in \mathbb{N}^n$; and a surround value $v \in \{-1, 1\}$.

Output: A TIR $\mathbf{T}$ for A, with those parameters.

```
1  T = NEW-IMG-ARRAY(N, {−1, 0, +1})
2  for each p ∈ N̂ do
3  |  c̃ = T^□[p]
4  |  x̃ = h̃(c)
5  |  if x̃.hi ≤ 0 then
6  |  |  T[p] = −1
7  |  else if x̃.lo ≥ 0 then
8  |  |  T[p] = +1
9  |  else
10 |  |  T[p] = 0
11 return (T, σ, v)
```

Chapter 5

Conversion of simple images to BMR

In this chapter we describe algorithms to extract a boundary mesh representation (BMR) from simple image representations — binary and ternary.

5.1 Conversion from BIR

Recall that the boundary of $\partial\mathbf{I}$ of a BIR $\mathbf{I}$ was defined as the union of all cell faces that are adjacent to at least one cell of value 0 and at least one cell of value 1. Therefore, a straightforward algorithm to construct a BMR $\mathbf{B}$ from a BIR $\mathbf{I} = (\mathtt{I}, \sigma, v)$ is to enumerate each cell $\mathbf{I}^{\square}[p]$ such that $\mathbf{I}[p] = 1 - v$, and inspect the $2n$ cells $\mathbf{I}[q] = \mathbf{I}[p \pm \mathbf{e}_i]$ that share an $(n-1)$-dimensional face with it. For every such pair with $\mathbf{I}[q] = v$, the shared face is added to the BMR $\mathbf{T}$. See Algorithm 10. The function `DECOMP-FACE` decomposes that face into a set of one or more $(n-1)$-simplices.

Algorithm 10: `BIN-TO-BOUNDARY`

Input: An n-dimensional BIR $\mathbf{I} = (\mathtt{I}, \sigma, v)$ with domain $\hat{N}$
Output: A BMR $\mathbf{B}$ that describes $\partial\mathbf{I}$

```
1  B = {}
2  for each p ∈ N̂ do
3  |  if I[p] = 1 − v then
4  |  |  for each i ∈ {0, 1, ..., n − 1} do
5  |  |  |  for each s ∈ {−1, +1} do
6  |  |  |  |  q = p + s e_i
7  |  |  |  |  if I[q] = v then
8  |  |  |  |  |  H = DECOMP-FACE(p, q)
9  |  |  |  |  |  B = B ∪ H
10 return B = (B, 1 − 2v)
```

5.2 Conversion from TIR

The direct conversion from a ternary image representation $\mathbf{T}$ to boundary representation is complicated because $\partial\mathbf{T}$ is not an $(n-1)$-manifold, but is a "fat" set with non-empty

interior.

The required result would be a set of $(n-1)$-dimensional simplices that are contained in $\partial\mathbf{T}$, intersect properly (only at their lower-dimensional faces), and constitute an $(n-1)$-dimensional borderless orientable manifold that separates any point in $\mathrm{int}(\mathbf{T})$ from any point in $\mathrm{ext}(\mathbf{T})$.

Several strategies are possible for creating such a mesh. The simplest one is to convert the ternary representation to a binary one by expanding the sets of cells with value $+1$ and -1 until all cells with value 0 are eliminated.

This goal can be achieved by successive passes of digital dilation. In each pass, two sets of indices $P_+, P_- \subseteq \mathbb{Z}^n$ are identified, corresponding to the cells of $\mathbf{T}$ that have value 0 and are adjacent to cells with value $+1$ and -1, respectively. Then all cells $\mathbf{T}[p]$ with $p \in P_+ \cup P_-$ are set to $+1$ if $p \in P_+ \setminus P_-$, to -1 if $p \in P_- \setminus P_+$, and randomly to -1 or $+1$ if $p \in P_+ \cap P_-$.

Part III

The clipped signed distance representation

Chapter 6

Clipped Euclidean distance representation (CDR)

In this chapter, we will define the main subject of this dissertation and its intended semantics, and present some of its mathematical properties.

6.1 Definition

The (*discrete*) *clipped* (*signed Euclidean*) *distance representation* (CDR), which is the main subject of this dissertation, is an image representation $\mathbf{K} = (\mathtt{K}, \sigma, v, \gamma, m)$, where γ is a floating point value, m is a positive integer, $\mathtt{K}$ is a n-dimensional image array with values $V = \{-m, \ldots, +m\}$, and v is the surround value which must be either $-m$ or $+m$.

The idea is that each value $\mathbf{K}[p]$ is the signed Euclidean distance $d^+(c, A)$ for a shape A, clipped to the maximum absolute value $\delta = \gamma m$, and quantized with step γ; where c is the center of the cell $\mathbf{K}^\square[p]$.

We assume that each cell value u gives only a lower bound to the unsigned distance from the cell's center c to the boundary of A, with the sign $+$ or $-$ depending on whether c is in the exterior or the interior of A, respectively. Specifically,

$$\left|d^+(c, A)\right| \le \mathrm{rad}^{\lfloor\circ\rfloor}(u, m, \gamma) \tag{6.1}$$

where

$$\mathrm{rad}^{\lfloor\circ\rfloor}(u, m, \gamma) = \begin{cases} \gamma\left(u - \frac{1}{2}\right) & \text{if } u > 0 \\ -\gamma\left(u + \frac{1}{2}\right) & \text{if } u < 0 \\ 0 & \text{if } u = 0. \end{cases} \tag{6.2}$$

Accordingly, we define

$$\mathbf{K}^{\lfloor\mathsf{R}\rfloor}[p] = \mathrm{rad}^{\lfloor\circ\rfloor}(\mathbf{K}[p], m, \gamma). \tag{6.3}$$

Since u is assumed to give only a lower bound, we refer to this semantics of the CDR as the *weak interpretation*.

6.2 The union-of-balls interpretation

According to Equation (6.1) the value $u = \mathbf{K}[p]$ says that the open ball with radius $\mathbf{K}^{\lfloor \mathsf{R} \rfloor}[p]$ centered at $\mathbf{K}^{\boxdot}[p]$ is entirely inside the object A if $\mathbf{K}[p] < 0$, or entirely outside A if $\mathbf{K}[p] > 0$. If $\mathbf{K}[p] = 0$ then the representation says nothing about the state of the cell $\mathbf{K}^{\square}[p]$ relative to A. See Figures 6.1 – 6.4.

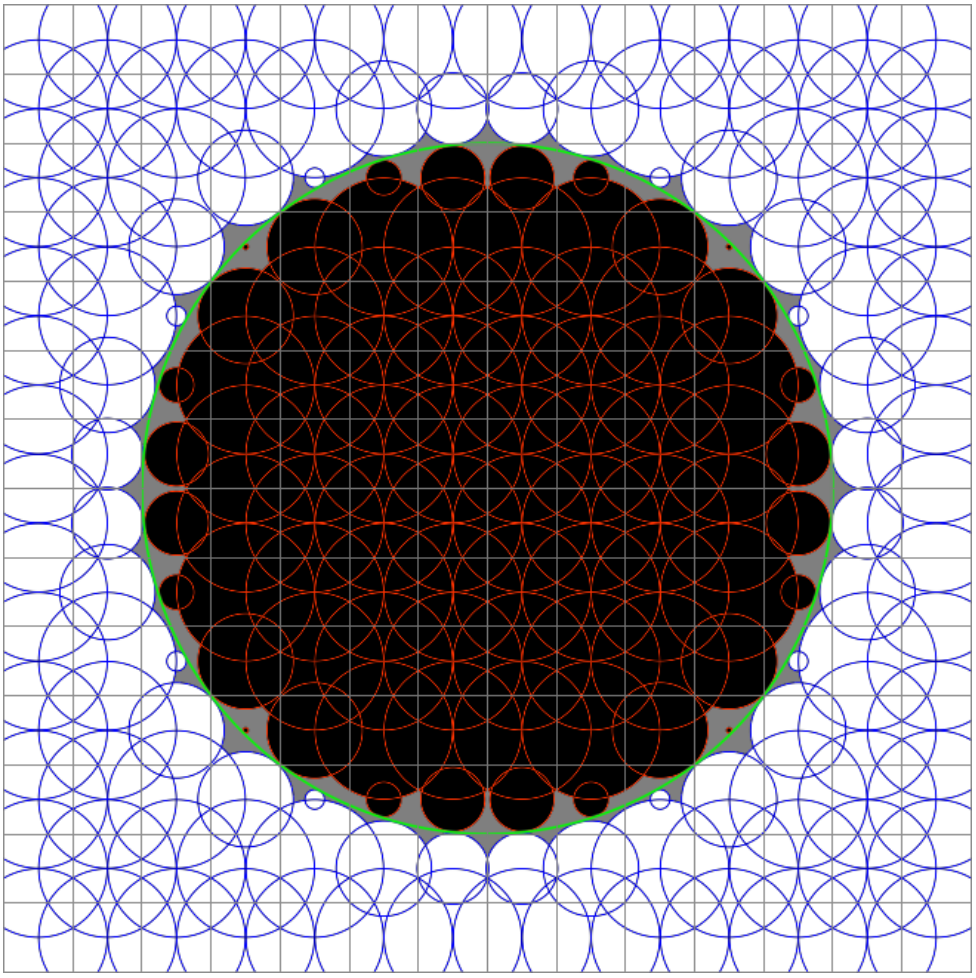


Figure 6.1: Clipped distance representation of a disk with radius 5 centered at $(8, 8)$ (green outline), with cell size $\sigma = \delta = 1$, $m = 127$ (so that $\gamma = 1/127$). All dimensions are in mm. The image size is 16×16; the grid lines show the pixel boundaries.

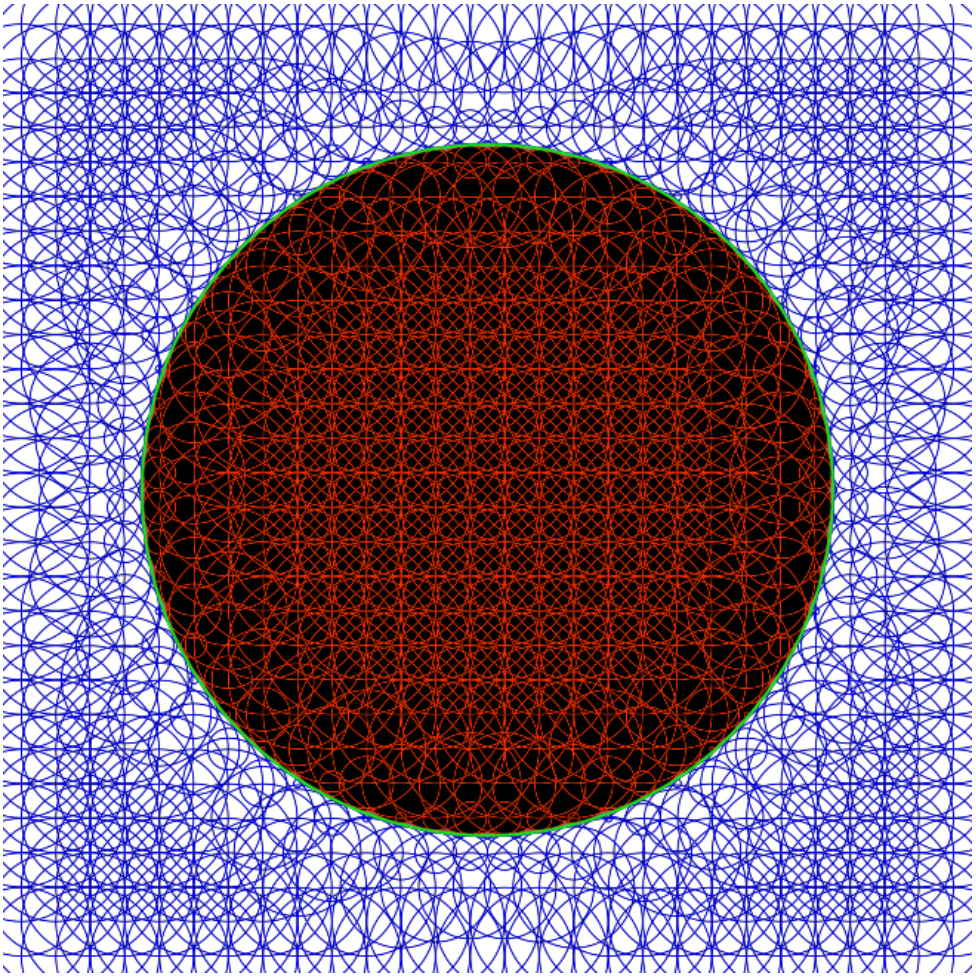


Figure 6.2: Clipped distance representation of a disk with radius 5 centered at $(8, 8)$ (green outline), with cell size $\sigma = \delta = 1/2$, $m = 127$ (so that $\gamma = 1/254$). All dimensions are in mm. The image size is 32×32; the grid lines show the pixel boundaries.

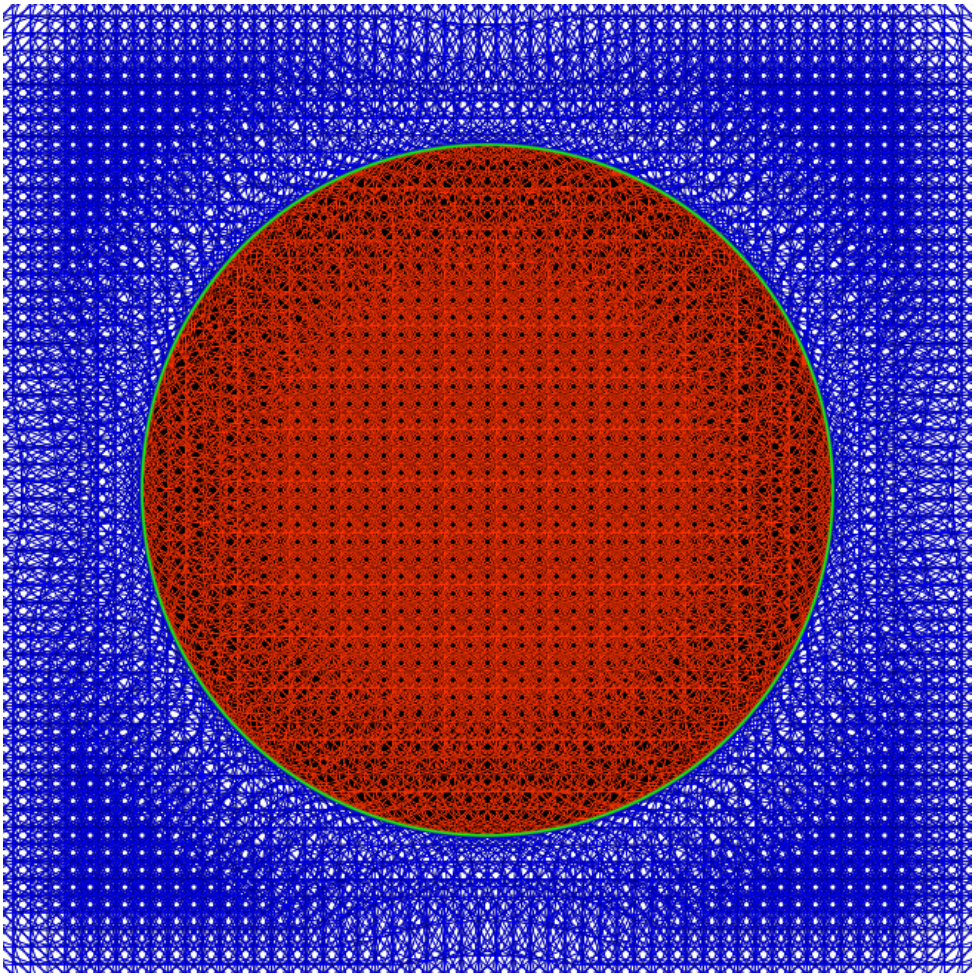


Figure 6.3: Clipped distance representation of a disk with radius 5 centered at $(8, 8)$ (green outline), with cell size $\sigma = \delta = 1/4$, $m = 127$ (so that $\gamma = 1/508$). All dimensions are in mm. The image size is 64×64; the grid lines show the pixel boundaries.

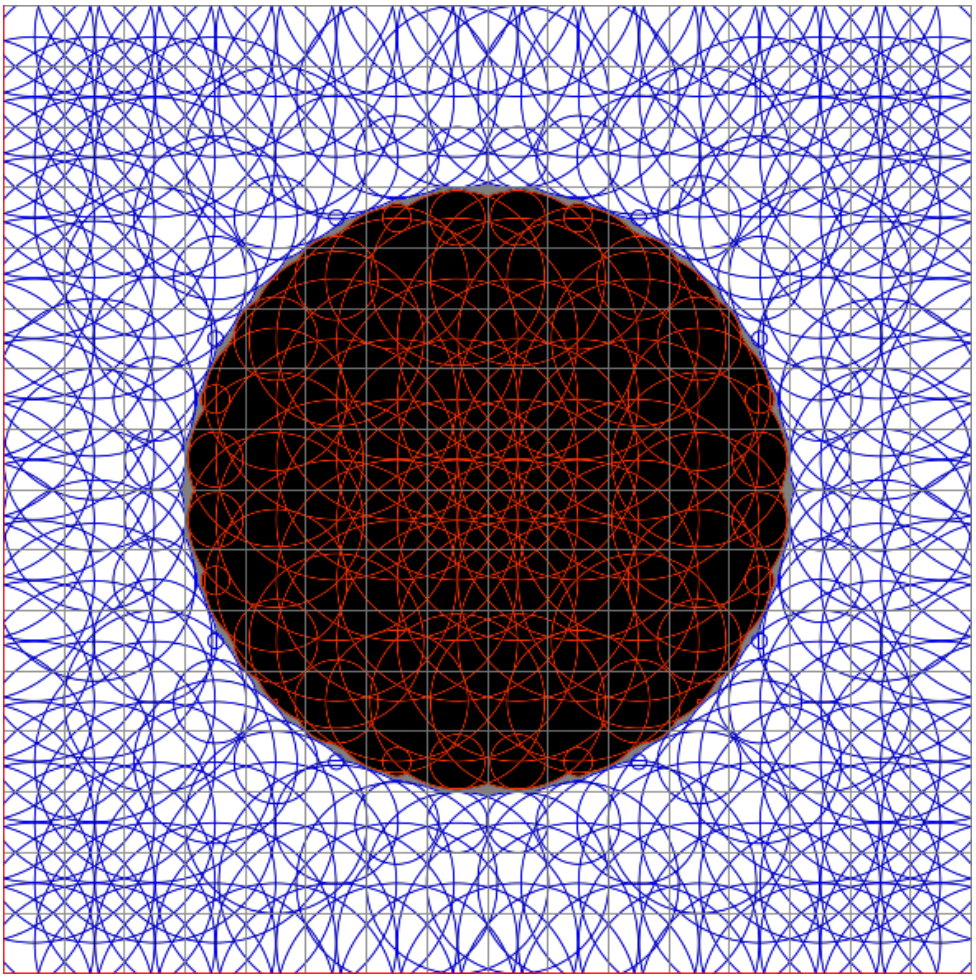


Figure 6.4: Clipped distance representation of a disk with radius 5 centered at $(8, 8)$ (green outline), with cell size $\sigma = 1$, $\delta = 2\sigma = 2$, $m = 127$ (so that $\gamma = 2/127$). All dimensions are in mm. The image size is 16×16; the grid lines show the pixel boundaries. Compare with Figure 6.1; note that the larger δ improves the accuracy of the boundary.

We denote by $\mathbf{K}^{\lfloor\circ\rfloor}[p]$ the open ball centered at $\mathbf{K}^{\boxdot}[p]$ with radius $\mathbf{K}^{\lfloor\mathsf{R}\rfloor}[p]$. That is

$$\mathbf{K}^{\lfloor\circ\rfloor}[p] = \operatorname{int}\left(\mathbf{K}^{\lfloor\mathsf{R}\rfloor}[p]\mathsf{B}^n + \mathbf{K}^{\boxdot}[p]\right). \tag{6.4}$$

Note that if $\mathbf{K}[p] = 0$ then $\mathbf{K}^{\lfloor\circ\rfloor}[p]$ is empty.

In the weak interpretation, therefore, we define the *interior*, *exterior*, and *boundary* of the CDR $\mathbf{K}$ as

$$\operatorname{int}(\mathbf{K}) = \bigcup_{\substack{p\in\mathbb{Z}^n \\ \mathbf{K}[p]>0}} \mathbf{K}^{\lfloor\circ\rfloor}[p], \tag{6.5}$$

$$\operatorname{ext}(\mathbf{K}) = \bigcup_{\substack{p\in\mathbb{Z}^n \\ \mathbf{K}[p]<0}} \mathbf{K}^{\lfloor\circ\rfloor}[p], \tag{6.6}$$

and

$$\partial(\mathbf{K}) = \mathbb{R}^n \setminus (\operatorname{int}(\mathbf{K}) \cup \operatorname{ext}(\mathbf{K})). \tag{6.7}$$

Therefore, as in the case of a proper TIR, a proper CDR $\mathbf{K}$ says that

$$\operatorname{int}(\mathbf{K}) \subseteq \operatorname{int}(A), \tag{6.8}$$

and

$$\operatorname{ext}(\mathbf{K}) \subseteq \operatorname{ext}(A). \tag{6.9}$$

It follows that

$$\partial(A) \subseteq \partial(\mathbf{K}). \tag{6.10}$$

Therefore a *correct* CDR must satisfy

$$\operatorname{int}(\mathbf{K}) \cap \operatorname{ext}(\mathbf{K}) = \{\} \tag{6.11}$$

6.3 Correctness test algorithm

For convenience, we will use the notation $\mathcal{T}(\mathbf{K}, p)$ for the union of balls with conservative radius in the same side of the boundary as $\mathbf{K}^{\boxdot}[p]$, that is

$$\mathcal{T}(\mathbf{K}, p) = \begin{cases} \operatorname{int}(\mathbf{K}) & \text{if } \mathbf{K}[p] < 0 \\ \operatorname{ext}(\mathbf{K}) & \text{if } \mathbf{K}[p] > 0 \\ \{\} & \text{if } \mathbf{K}[p] = 0 \end{cases} \tag{6.12}$$

and, complementary

$$\overline{\mathcal{T}}(\mathbf{K}, p) = \begin{cases} \operatorname{ext}(\mathbf{K}) & \text{if } \mathbf{K}[p] < 0 \\ \operatorname{int}(\mathbf{K}) & \text{if } \mathbf{K}[p] > 0 \\ \{\} & \text{if } \mathbf{K}[p] = 0. \end{cases} \tag{6.13}$$

The correctness test algorithm depend on the fact that a CDR $\mathbf{K}$ is correct if every point $p \in \mathbb{Z}^n$ we have

$$\mathbf{K}^{\lfloor\circ\rfloor}[p] \cap \overline{\mathcal{T}}(\mathbf{K}, p) = \{\}\,, \tag{6.14}$$

and we say that $\mathbf{K}$ is correct if every point $p \in \mathbb{Z}^n$ is correct. Given a CDR $\mathbf{K}$ we want to know if a point p is correct. To check the condition in Equation (6.14), we only have to look the points in $\overline{\mathcal{T}}(\mathbf{K}, p)$ in the neighborhood

$$\mathcal{N}(\mathbf{K}, p) = \left\{i \in \mathbb{Z}^n : \left\|\mathbf{K}^{\boxdot}[i] - \mathbf{K}^{\boxdot}[p]\right\| \leq \delta + \mathbf{K}^{\lfloor\mathsf{R}\rfloor}[p]\right\} \tag{6.15}$$

because the maximum radius of the balls that composes $\overline{\mathcal{T}}(\mathbf{K}, p)$ is δ, therefore, there is no need to check balls centered at a greater distance than $\delta + \mathbf{K}^{\lfloor\mathsf{R}\rfloor}[p]$. The direct algorithm is to look for points $\overline{\mathcal{T}}(\mathbf{K}, p)$ in this neighborhood and check the sphere intersection. This algorithm is in $O(|\mathcal{N}(\mathbf{K}, p)|) = O((4\delta/\sigma)^n)$, and the worst case is when $\mathbf{K}^{\lfloor\mathsf{R}\rfloor}[p] = \delta$.

6.4 Tightness

Recall that the value of a CDR cell gives only a lower bound to the distance from the center of that cell to the boundary of the shape. We say that $\mathbf{K}$ is *tight at* a point $p \in \mathbb{Z}^n$ if that lower bound is tight, that is, if $\mathbf{K}[p]$ cannot be replaced by any value with greater magnitude without changing $\text{int}(\mathbf{K})$ or $\text{ext}(\mathbf{K})$. In other words, $\mathbf{K}$ is tight at p if

$$|\mathbf{K}[p]| = \max\left\{\, r : 0 \leq r \leq m \wedge \text{int}\left(\text{rad}^{\lfloor\circ\rfloor}(r, m, \gamma)\mathsf{B}^n + \mathbf{K}^{\boxdot}[p]\right) \subseteq \mathcal{T}(\mathbf{K}, p)\,\right\}. \tag{6.16}$$

Note that $\mathbf{K}$ is always tight at p if $\mathbf{K}[p] = 0$ or $|\mathbf{K}[p]| = m$. If $\mathbf{K}$ is not tight at p, we say that it is *loose* at p.

We say that $\mathbf{K}$ is *tight* if it is tight at every $p \in \mathbb{Z}^n$, and *loose* otherwise.

Tightness is a useful property because it means that $\mathbf{K}[p]$ contain the maximum possible information about the distance from $\mathbf{K}^{\boxdot}[p]$ to ∂A that can be deduced from all cell values of $\mathbf{K}$, and encoded in the m, γ quantization.

6.4.1 Tightness test algorithm

We now describe an algorithm to compute the right hand side of Equation (6.16), in other words, the test if the lower bound ball is inside $\mathcal{T}(\mathbf{K}, p)$. For the rest of this Section we assume that $\mathbf{K}$ is correct in the sense of the Section 6.3.

The algorithm will check the tightness for each cell of $\mathbf{K}$, we can reuse computations of pixels of the same connected component. The mentioned connected component of each cell uses the information that there is no need to check the distances bigger than δ.

First we replace the $\mathcal{T}(\mathbf{K}, p)$ by a smaller subset without changing the result of Equation (6.16). Given a point $p \in \mathbb{Z}^n$ such that $\mathbf{K}[p] \neq 0$ and $|\mathbf{K}[p]| < m$ we define the neighborhood $\mathcal{N}^\star$ where the lower bound balls intersects, that is

$$\mathcal{N}^\star(\mathbf{K}, p) = \left\{i \in \mathbb{Z}^n : \mathbf{K}^{\lfloor\circ\rfloor}[p] \cap \mathbf{K}^{\lfloor\circ\rfloor}[i] \neq \{\}\right\}. \tag{6.17}$$

Then the *relevant* cell indices are

$$\mathcal{N}(\mathbf{K}, p) = \left\{ i \in C(\mathbf{K}, \mathcal{N}^\star, p) : \left\| \mathbf{K}^{\boxdot}[p] - \mathbf{K}^{\boxdot}[i] \right\| \leq 2\delta \right\} \tag{6.18}$$

where $C(\mathbf{K}, \mathcal{N}^\star, p)$ is the connected component given the neighborhood $\mathcal{N}^\star$ and the point p as seed. We reduce the tightness test problem as the problem of computing the distance from the center of the cell $\mathbf{K}^{\boxdot}[p]$ to the boundary of the union of balls

$$\mathcal{B}(\mathbf{K}, p) = \left\{ \mathrm{cl}\left(\mathbf{K}^{\lfloor \circ \rfloor}[i] \right) : i \in \mathcal{N}(\mathbf{K}, p) \right\}. \tag{6.19}$$

Once we have the distance $w = d(\mathbf{K}^{\boxdot}[p], \partial(\mathcal{B}(\mathbf{K}, p)))$, $\mathbf{K}$ is tight at p if inequality $\mathbf{K}^{\lfloor \mathsf{R} \rfloor}[p] \leq w \leq \mathbf{K}^{\lceil \mathsf{R} \rceil}[p]$ holds. Note that, for $p, q \in \mathcal{N}^\star(\mathbf{K}, p)$ the union of balls $\mathcal{B}(\mathbf{K}, p)$ can be different from $\mathcal{B}(\mathbf{K}, q)$. In spite of $\mathcal{N}^\star(\mathbf{K}, p)$ is always equal to $\mathcal{N}^\star(\mathbf{K}, q)$ the neighborhoods $\mathcal{N}(\mathbf{K}, p)$ and $\mathcal{N}(\mathbf{K}, q)$ can differ.

Algorithm 11 computes the *weak distance transform* given a CDR $\mathbf{K}$. The weak distance transform is a image $\mathtt{D}$ of float values with same domain size than $\mathbf{K}$ where each cell $\mathtt{D}[p]$ is equal to the distance from the center of the cell $\mathbf{K}^{\boxdot}[p]$ to the boundary of $\mathcal{T}(\mathbf{K}, p)$.

Given a CDR $\mathbf{K}$ and a neighborhood $\mathcal{N}$, `GET-CONNECTED-COMPONENTS` returns a tuple comprised of a image array $\mathtt{C}$ and a integer k. The integer k is the number of connected component in $\mathbf{K}$ given the neighborhood $\mathcal{N}$ and the image $\mathtt{C}$ has values in $\{0, \ldots, k-1\}$ and for each position $\mathtt{C}[p]$ the value is associated with the component label.

After computing the weak distance transform $\mathtt{D}$ of $\mathbf{K}$, we can check if $\mathbf{K}$ is tight at a point p with the inequality $\mathbf{K}^{\lfloor \mathsf{R} \rfloor}[p] \leq \mathtt{D}[p] \leq \mathbf{K}^{\lceil \mathsf{R} \rceil}[p]$.

The theory and algorithm for the function `DISTANCE-TO-BOUNDARY` is discussed in Appendix B. For this usage, to avoid recalculations we could implement Algorithm 17 with some memoization technique on the object candidates of each connected component.

Algorithm 11: `WEAK-DISTANCE-TRANSFORM`

Input: A CDR $\mathbf{K}$ with domain $\hat{N}$
Output: A image $\mathtt{D}$ of floats that maps each index i to the distance $d(\mathbf{K}^{\boxdot}[i], \mathcal{T}(\mathbf{K}, i))$

```
1 N = the neighborhood defined in Equation (6.18)
2 C, k = GET-CONNECTED-COMPONENTS(K, N)
3 B =new array with size k of sets of balls
4 for each i in N̂ do
5 |   B[C[i]] = B[C[i]] ∪ K^⌊∘⌋[i]
6 for each i in N̂ do
7 |   d = DISTANCE-TO-BOUNDARY(B[C[i]], K^⊡[i])
8 |   D[i] = min(d, δ)
9 return D
```

6.5 Accuracy

The maximum error made when converting $d_\delta^+(\mathbf{K}^\boxdot[p], A)$ to an integer is half of the difference between consecutive values, that is, $\delta/(2m)$. Note that the domain box of $\mathbf{K}$ must contain A or $\overline{A}$ with a sufficient margin – specifically, at least $\lceil \delta/\sigma \rceil$ cells on every side.

Each element of `K` can be stored as a b-bit signed integer where $b = \lceil \lg(2m+1) \rceil$. The total storage size of the image is therefore $b \prod_i N_i$. Again, a quadtree or octree data structure may reduce this number considerably if A has a sufficiently simple shape. The storage size of the CDR is the size of a BIR `I` times the factor b/k^n. Thus, for example, for $m = 127$ and $k = 16$, we have $b = 8$, and the storage size is reduced by a factor of $1/32$ for $n = 2$, and $1/512$ for $n = 3$.

Chapter 7

Conversion to CDR

In this chapter we discuss some fundamental algorithms to convert common shape representations to the clipped signed distance representation (CDR).

7.1 Conversion from IDR

Algorithm 12 constructs the CDR of a shape B from a procedural representation of it; specifically, from an interval-arithmetic implementation of the signed Euclidean distance function $h(p) = d^+(p,\mathrm{A})$.

Algorithm 12: `PROCEDURAL-TO-CDR`

Input: An IDR $\widetilde{h} : \mathbb{I}^n \to \mathbb{I}$ of a target object A; a positive cell size $\sigma \in \mathbb{F}$; a domain size $N \in \mathbb{N}^n$; a positive distance quantization step $\gamma \in \mathbb{F}$; a positive integer m; and a surround value $v \in \{-m, +m\}$.

Output: A CDR $\mathbf{K}$ for A, with those parameters.

$\delta = m\gamma$
$\mathtt{K} = \mathtt{NEW\text{-}IMG\text{-}ARRAY}(N, \{-m, \ldots, +m\})$
for each $p \in \widehat{N}$ **do**
  $c = \sigma\left(p + \frac{1}{2}\mathbf{1}^n\right)$
  $\widetilde{x} = \widetilde{h}(c)$
  if $\widetilde{x}.hi \leq -\delta$ **then**
    $\mathtt{K}[p] = -m$
  else if $\widetilde{x}.lo \geq +\delta$ **then**
    $\mathtt{K}[p] = +m$
  else
    $\widetilde{t} = \widetilde{x}/\gamma$
    if $\widetilde{x}.hi \leq 0$ **then**
      $\mathtt{K}[p] = \lceil \widetilde{t}.hi \rceil$
    else if $\widetilde{x}.lo \geq 0$ **then**
      $\mathtt{K}[p] = \lfloor \widetilde{t}.lo \rfloor$
    else
      $\mathtt{K}[p] = 0$
return $(\mathtt{K}, \sigma, v, m, \gamma)$

See Figure 7.1 for an example of output.

Note that the real value $h(c)/\gamma$ is rounded towards 0 in Lines 13 and 15, in order to ensure that the weak interpretation of the CDR *repK* is consistent with the shape A, as described by $\widetilde{h}$.

The tests in Lines 6 and 8 could be removed by rewriting Line 13 as $\mathtt{K}[p] = \max\{-m, \lceil \widetilde{t}.hi \rceil\}$ and $\mathtt{K}[p] = \min\{+m, \lfloor \widetilde{t}.hi \rfloor\}$, respectively. However, the version shown avoids computing the IA division $\widetilde{t} = \widetilde{x}/\gamma$ in the common case when c is more than δ away from the boundary of A.

In this algorithm we have ignored possible rounding errors in the computation of δ. If γ is a binary fraction $g/2^s$ with $g, s \in \mathbb{Z}$, and $|g|$, $|s|$, and m are small enough (say, $\gamma = 375/1024$ and $m = 127$), then the floating-point multiplication $m\gamma$ will be exact. Also, for any value of N likely to occur in practice, the floating-point computation of c will be exact.

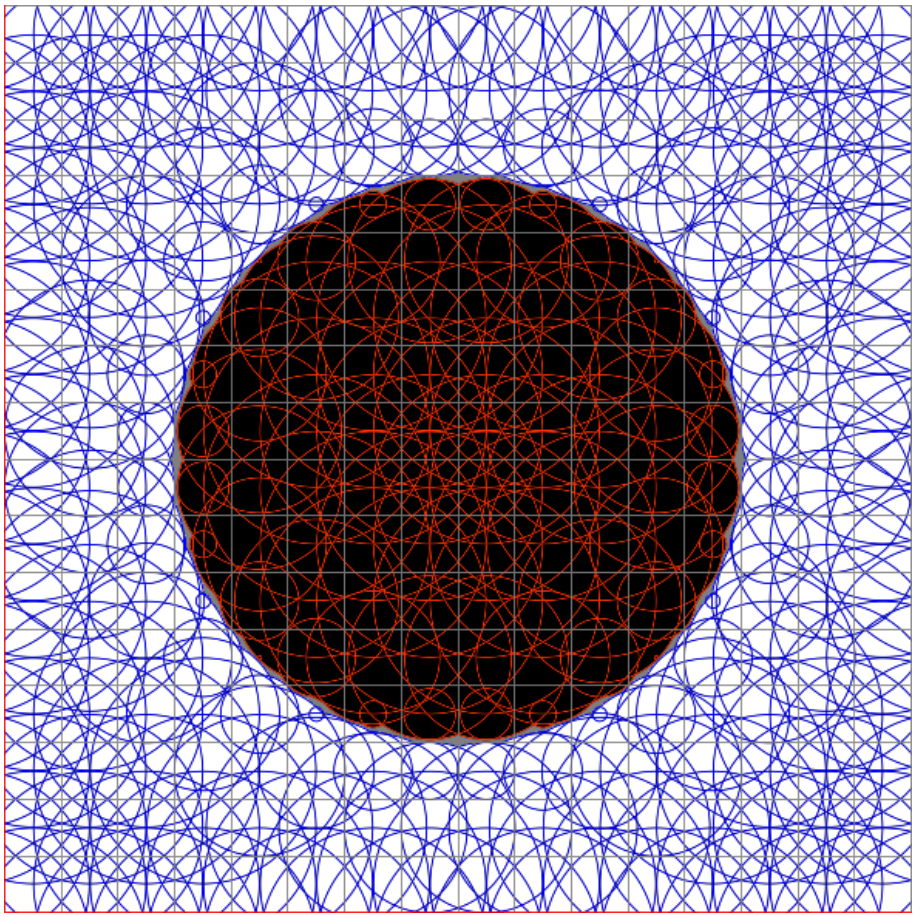


Figure 7.1: Clipped distance representation of a disk of radius 5 centered at $(8, 8)$ (green outline), created from the corresponding IDR by Algorithm 12, with $\sigma = 1$, $\delta = 2\sigma = 2$, $m = 127$ (so that $\gamma = 2/127$), $N = (16, 16)$. All dimensions are in mm. The grid lines show the pixel boundaries.

7.2 Conversion from BIR

We now consider the conversion of an n-dimensional BIR $\mathbf{I} = (\mathtt{I}, \sigma, v)$ to a discrete clipped distance representation $\mathbf{K} = (\mathtt{K}, k\sigma, v', \gamma, m)$, for some integer $k \geq 1$, with specified distance quantization parameters $m \geq 1$ and $\gamma > 0$. Note that each cell of $\mathbf{K}$ is an n-dimensional hypercube of side $k\sigma$, that is the union of k^n cells of $\mathbf{I}$. The resulting CDR is meant represent, as accurately as possible, the shape $\mathrm{int}(\mathbf{I})$, the interior of the union of all (closed) cells of $\mathbf{I}$ which have value 1. The surround value v' of $\mathbf{K}$ will be $+m$ if $v = 0$, and $-m$ if $v = 1$.

For simplicity, we assume that the size N_i of $\mathtt{I}$ along any axis is a multiple of k, and that the corresponding size M_i of $\mathtt{K}$ can be set to N_i/k. Note that a voxel $\mathbf{K}[p]$ of the resulting CDR will be non-trivial (neither $-m$ nor $+m$) only if the distance from its center to the boundary of A is less than $\delta = m\gamma$. Therefore, the above assumption requires that $\mathtt{I}$ has enough layers of exterior cells (all with value 0) on all sides, so that all non-trivial pixels of $\mathbf{K}$ will lie inside the new domain $\widehat{N}'$. Specifically, any cell of $\mathbf{I}$ with value 1 must be separated from any point of the boundary of $\mathcal{B}\mathbf{I}$ by at least $\lceil \delta/\sigma \rceil$ cells with value 0.

The basic idea of the algorithm is to set each element $\mathtt{K}[q]$ whose cell center $\mathbf{K}^{\boxdot}[q]$ is in $\mathrm{ext}(\mathbf{I})$ to the smallest distance between c and any cell $\mathbf{I}^{\square}[p]$ of $\mathbf{I}$ that has value 1 — quantized with step γ, rounded towards 0, and clipped to maximum value $+m$. Elements whose center c is in $\mathrm{int}(\mathbf{I})$ are treated symmetrically. If c is on $\partial\mathbf{I}$ (which may happen only if k is even), then $\mathtt{K}[q]$ must be set to 0.

With the above assumptions, we only need to perform this analysis for the indices q in the domain $\widehat{M}$ of $\mathtt{K}$. Moreover, for each such q, we need to consider only those indices p such that $\mathrm{dist}(c, D) < \delta$, where

$$c = \mathbf{K}^{\boxdot}[q] = k\sigma(q + (1/2)\mathbf{1}^n) \tag{7.1}$$

and

$$D = \mathbf{I}^{\square}[p] = \sigma(\mathsf{K}^n + p + (1/2)\mathbf{1}^n). \tag{7.2}$$

That is, we need to consider only those index vectors p such that

$$\mathrm{dist}(k(q + (1/2)\mathbf{1}^n), \mathsf{K}^n + p + (1/2)\mathbf{1}^n) < \delta/\sigma \tag{7.3}$$

which is equivalent to

$$(\sigma/\gamma)\,\mathrm{dist}((k/2)\mathbf{1}^n, \mathsf{K}^n + p - kq + (1/2)\mathbf{1}^n) < m \tag{7.4}$$

This condition depends only on the integer vector $r = p - kq$. Therefore, let's define the function

$$d(r) = \lfloor (\sigma/\gamma)\,\mathrm{dist}((k/2)\mathbf{1}^n, \mathsf{K}^n + r + (1/2)\mathbf{1}^n) \rfloor \tag{7.5}$$

We precompute the set of displacement vectors

$$\mathcal{N} = \{\, r \in \mathbb{Z}^n : d(r) < m \,\} \tag{7.6}$$

and save the corresponding values of $d(r)$. The set $\mathcal{N}$ is the set of all indices p of $\mathbf{I}$ that

need to be considered when computing the "lowest" voxel $\mathbf{K}[\mathbf{0}^n]$.

The set $\mathcal{N}$ contains, in particular, the vector $r^* = \lfloor k/2 \rfloor \mathbf{1}^n$, such that the point $\mathbf{K}^{\boxdot}[\mathbf{0}^n] = (k\sigma/2)\mathbf{1}^n$ is in the interior (if k is odd) or on the boundary (if k is even) of the cell $\mathbf{I}^{\square}[r^*]$. Either way, $d(r^*)$ is 0.

The following steps are then repeated for each $q \in \mathcal{D}\,K$. We fetch the value $u = \mathtt{I}[kq + r^*]$. Note that if $u = 0$ then $c \notin \mathrm{int}(\mathbf{I})$, and if $u = 1$ then$c \notin \mathrm{ext}(\mathbf{I})$. Then we compute

$$x = \min \{\, d(r) : r \in \mathcal{N} \wedge \mathtt{I}[kq + r] \neq u \,\} \tag{7.7}$$

except that we set $x = m$ if the set is empty. Then we set $\mathtt{K}[q]$ to $+x$ if $u = 0$, or $-x$ if $u = 1$. See Algorithm 13.

Algorithm 13: `BINARY-TO-CDR`

Input: A BIR $\mathbf{I} = (\mathtt{I}, \sigma, v)$; a positive coarsening factor $k \in \mathbb{N}$; a positive quantization step $\gamma \in \mathbb{F}$; and a positive integer m.

Output: A CDR $\mathbf{K}$ for $\mathrm{cl}(\mathrm{int}(\mathbf{I}))$, with cell size $k\sigma$ and parameters m, γ.

$N = \mathcal{D}(\mathtt{I})$; $M = N/k$
$\mathtt{K} = \mathtt{NEW\text{-}IMG\text{-}ARRAY}(M, \{-m, \ldots, +m\})$
$\mathcal{N} = \{\, r \in \mathbb{Z}^n : d(r) < m \,\}$
$r^* = \lfloor k/2 \rfloor \, \mathbf{1}^n$
for each $q \in \widehat{M}$ **do**
  $u = \mathtt{K}[kq + r^*]$
  $x = m$
  for each $r \in \mathcal{N}$ **do**
    if $\mathtt{I}[kq + r] = 1 - u$ **then** $x = \min\{x, d(r)\}$
  $\mathtt{K}[p] = (1 - 2u)x$
return $(\mathtt{K}, k\sigma, (1 - 2v)m, m, \gamma)$

The running time of this algorithm, in the worst case, is dominated by the inner loop, which is executed $|\mathcal{N}|\,|\mathcal{D}(\mathbf{K})| = |\mathcal{N}|\,|\mathcal{D}(\mathbf{I})|\,/k^n$. The neighborhood size $|\mathcal{N}|$ is approximately $\mathcal{V}_n(\delta/\sigma)^n$ where $\mathcal{V}_n$ is the hypervolume of the ball of unit radius of $\mathbb{R}^n$, which is π for $n = 2$ and $4\pi/3$ for $n = 3$. Note that δ must be at least $(k\sigma/2)\sqrt{n}$ to obtain a meaningful CDR; in that case, the inner loop is executed at least $|\mathcal{D}(\mathbf{I})|$ times.

The running time can be improved in several ways. First, the list $\mathcal{N}$ of displacement vectors should be sorted in order of increasing $d(r)$. That way, the computation of x can be halted as soon as one element r with $\mathtt{I}[kq + r] \neq u$ is found.

The algorithm can be further improved by quickly detecting most index vectors q where $\mathrm{dist}(\mathbf{K}^{\boxdot}[q], \partial A)$ is easily determined to be more than δ. That can be achieved by computing a TIR $\mathbf{J}$ with cell size $k\sigma$ and same domain as $\mathbf{K}$, with the property that $\mathrm{int}(\mathbf{J}) \subseteq \mathrm{int}(\mathbf{I})$ and $\mathrm{ext}(\mathbf{J}) \subseteq \mathrm{ext}(\mathbf{I})$. Then $\mathbf{J}$ is modified by digitally eroding the sets $\mathrm{int}(\mathbf{J})$ and $\mathrm{ext}(\mathbf{J})$ as needed. Then, if $\mathtt{J}[q]$ is ± 1, we can safely set $\mathtt{K}[q]$ to $\pm m$.

7.3 Conversion from TIR

Now we consider the conversion of a ternary image representation $\mathbf{I}$ to a CDR $\mathbf{K}$. The goal is to have $\text{int}(\mathbf{K}) \subseteq \text{int}(\mathbf{I})$ and $\text{ext}(\mathbf{K}) \subseteq \text{ext}(\mathbf{I})$, with both sets as large as possible.

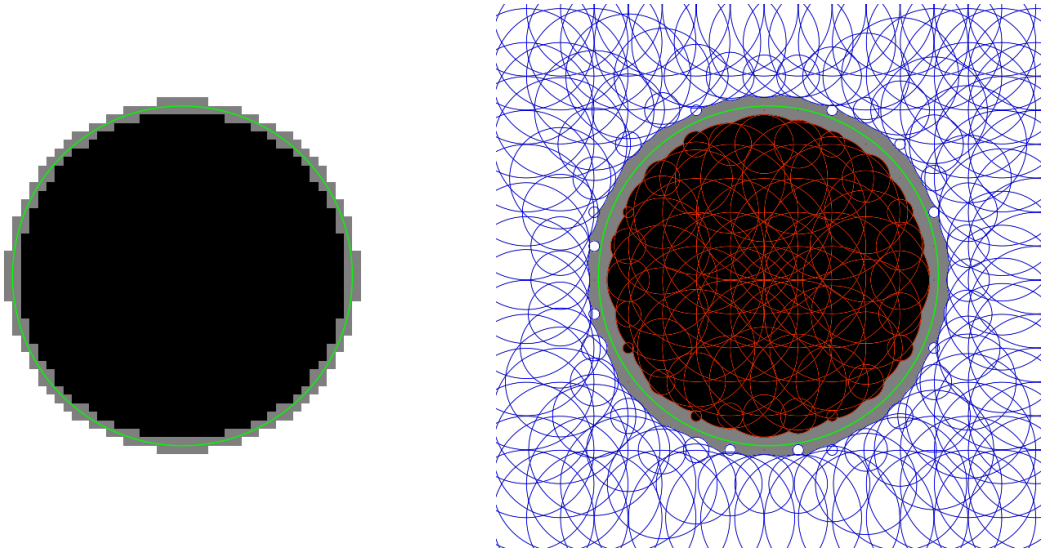
The conversion algorithm `TERNARY-TO-CDR()` (see Algorithm 14) is very similar to `BINARY-TO-CDR()` (Algorithm 13). The difference are mainly due to the encoding of interior and exterior as -1 and $+1$ instead of 1 and 0, respectively.

Algorithm 14: `TERNARY-TO-CDR`

Input: A TIR $\mathbf{I} = (\mathtt{I}, \sigma, v)$; a positive coarsening factor $k \in \mathbb{N}$; a positive quantization step $\gamma \in \mathbb{F}$; and a positive integer m.

Output: A CDR $\mathbf{K}$ with $\text{int}(\mathbf{K}) \subseteq \text{int}(\mathbf{I}))$, $\text{ext}(\mathbf{K}) \subseteq \text{ext}(\mathbf{I}))$, cell size $k\sigma$ and parameters m, γ.

$N = \mathcal{D}(\mathtt{I})$; $M = N/k$
$\mathtt{K} = \texttt{NEW-IMG-ARRAY}(M, \{-m, \ldots, +m\})$
$\mathcal{N} = \{ r \in \mathbb{Z}^n : d(r) < m \}$
$r^* = \lfloor k/2 \rfloor \, \mathbf{1}^n$
for each $q \in \widehat{M}$ **do**
  $u = \mathtt{K}[kq + r^*]$
  if $u \neq 0$ **then**
    $x = m$
    for each $r \in \mathcal{N}$ **do**
      if $\mathtt{I}[kq + r] = -u$ **then** $x = \min\{x, d(r)\}$
    $\mathtt{K}[p] = ux$
  else
    $\mathtt{K}[p] = 0$
return $(\mathtt{K}, k\sigma, vm, m, \gamma)$



(a) Input (b) Output

Figure 7.2: Conversion by Algorithm 14 of the ternary image representation **J** (left) of a disk of radius 5 centered at $(8, 8)$ (green outline) to a clipped distance representation **K** (right). The input TIR has cell size $\sigma = 1/4$ and image size 64×64. The output CDR has cell size $\sigma = 1$ (coarsening factor $k = 4$), image size 16×16, $m = 127$, and $\delta = 2\sigma = 2$ ($\gamma = 2/127$). All dimensions are in mm. The grid lines show pixel boundaries.

7.4 Changing the CDR parameters

In this section we consider the problem of changing the parameters of a CDR. Specifically, given a CDR $\mathbf{K}' = (\mathtt{K}', v', \sigma', m', \gamma')$ for some shape A, we want to produce another CDR $\mathbf{K} = (\mathtt{K}, v, \sigma, m, \gamma)$ for A, changing the parameters σ', m', and γ' to σ, m, and γ, respectively. This conversion also changes the saturation value $\delta' = m'\gamma'$ to $\delta = m\gamma$, and the surround value $v' = \pm m'$ to $v = \pm m$.

If the cell sizes σ' and σ have a large enough common divisor τ (that is, $\sigma' = r'\tau$ and $\sigma = r\tau$, for small positive integers r' and r), then a simple solution is to convert the given CDR $\mathbf{K}'$ to a TIR $\mathbf{I}$ with cell size τ or any sub-multiple thereof, using the procedure `CDR-TO-TERNARY()` (Algorithm 15); and then convert $\mathbf{I}$ into a CDR $\mathbf{K}$ with the desired parameters, with `TERNARY-TO-CDR()` (Algorithm 14). This method will work for any parameter changes.

If the cell sizes are the same, one may consider a direct *re-quantization* of the voxel values. Namely, we convert each voxel value $u' = \mathtt{K}'[p]$ to a (floating-point) distance value d using the parameter γ', then re-quantize d with parameters m, γ to obtain the new value $u = \mathtt{K}[p]$. That is,

$$u = \operatorname{sgn}(u') \min m, \lfloor\, |u'|\, \gamma'/\gamma \,\rfloor \tag{7.8}$$

where $\operatorname{sgn}(u') \in \{-1, 0, +1\}$ is the sign of u'.

Note that this formula reduces to $u = \max\{-m, \min\{+m, u'\}\}$ if the quantization step is not changed ($\gamma = \gamma'$). More generally, $u = \max\{-m, \min\{+m, su'\}\}$ if the old step γ' is an integer multiple $s\gamma$ of the new one. Otherwise, the formula $|u'|\,\gamma'/\gamma$ must be computed with floating-point rounding mode set to "towards zero".

For efficiency, Equation (7.8) can be pre-computed for all values u' in $\{-m', \ldots, +m'\}$. Then the re-quantization reduces to a table lookup.

This simple re-quantization may be sufficient if the *new* step is an integer multiple of the old one ($\gamma/\gamma' \in \mathbb{Z}$), and the clipping distance is not to be increased ($\delta \le \delta'$). Then voxel values that are tight in the given image (see Section 6.4) will remain tight in the new one. In particular, if $\mathtt{K}'[p]$ is $\pm m'$, then $\mathtt{K}[p]$ will be $\pm m$. Then, if $\mathbf{K}'$ is self-consistent, $\mathbf{K}$ will be self-consistent too.

However, in general this operation will leave the new CDR $\mathbf{K}$ in a loose (not maximally informative) state. In particular, if $\delta > \delta'$, voxels that have absolute value m' in $\mathbf{K}'$ will have absolute value less than m in the new image $\mathbf{K}$ – even when very far from the boundary. Therefore, it may be appropriate, or even necessary, to apply a tightening algorithm to the resulting $\mathbf{K}$. See Section 6.4.

Chapter 8

Conversion from CDR to simple images

In this chapter we present algorithms to convert the CDR to simple image representations (BIR and TIR).

8.1 Conversion to TIR

First we describe how to convert a CDR $\mathbf{K} = (\mathtt{K}, \sigma, v, m, \gamma)$ of some shape A into a TIR $\mathbf{J} = (\mathtt{J}, \sigma', v')$ for the same shape. The goal is to ensure $\mathrm{int}(\mathbf{J}) \subseteq \mathrm{int}(\mathbf{K})$ and $\mathrm{ext}(\mathbf{J}) \subseteq \mathrm{ext}(\mathbf{K})$. If $\mathbf{K}$ is a correct representation of A, these conditions will ensure that $\mathbf{J}$ is too. The accuracy of $\mathbf{J}$ relative to $\mathbf{K}$ will depend on the new cell size σ'.

Procedure `CDR-TO-TERNARY` (Algorithm 15) is a straightforward algorithm for this conversion. It requires that the new cell size be a sub-multiple of the old one; that is, $\sigma' = \sigma/k$ for some small integer k. The procedure enumerates each voxel $\mathbf{K}[p]$ of the given CDR, and sets every voxel of $\mathbf{J}$ whose (closed) cell is entirely inside the corresponding (open) ball $\mathbf{K}^{\lfloor\circ\rfloor}[p]$ to $+1$ or -1, depending on whether $\mathbf{K}[p]$ is positive or negative, respectively. The surround value v' of $\mathbf{J}$ is of course set to $+1$ if the surround v of $\mathbf{K}$ is $+m$, and to -1 if v is $-m$.

For efficiency, Algorithm 15 pre-computes a list $\mathcal{N}_0, \mathcal{N}_1, \ldots, \mathcal{N}_m$ of neighborhoods, that describe the voxels of $\mathbf{J}$ that are inside the ball $\mathbf{K}^{\lfloor\circ\rfloor}[p]$, given $u = \mathbf{K}[p]$. Namely,

$$\mathcal{N}_u = \{\, r \in \mathbb{Z}^n : \ (\forall x \in \mathbf{J}^{\square}[kp + r]) \ \mathrm{dist}(\mathbf{K}^{\boxdot}[p], x) < u\gamma \,\} \tag{8.1}$$

This set does not depend on p, so it can be computed for $p = \mathbf{0}^n$. Since the distance from the point $c = \mathbf{K}^{\boxdot}[\mathbf{0}^n] = (\sigma/2)\mathbf{1}^n$ to a point x in a cell is maximum when x is a corner, we can write

$$\mathcal{N}_u = \{\, r \in \mathbb{Z}^n : g(r) < u\gamma \,\} \tag{8.2}$$

where $g(r)$ is the distance from c to the corner of $\mathbf{J}^{\square}[r]$ that is farthest from c, namely

$$g(r) = \max \{\, \mathrm{dist}(c, \sigma'(r + h)) : h \in \{0,1\}^n \,\} \tag{8.3}$$

These neighborhoods are nested, that is $\{\} = \mathcal{N}_0 \subseteq \mathcal{N}_1 \subseteq \cdots \subseteq \mathcal{N}_m$; and all the vectors r in $\mathcal{N}_u \setminus \mathcal{N}_{u-1}$ have $g(r)$ greater than those of $\mathcal{N}_{u-1}$, for every $u > 0$. Therefore, the algorithm needs to compute only the last neighborhood $\mathcal{N}^* = \mathcal{N}_m$, sort its vectors in

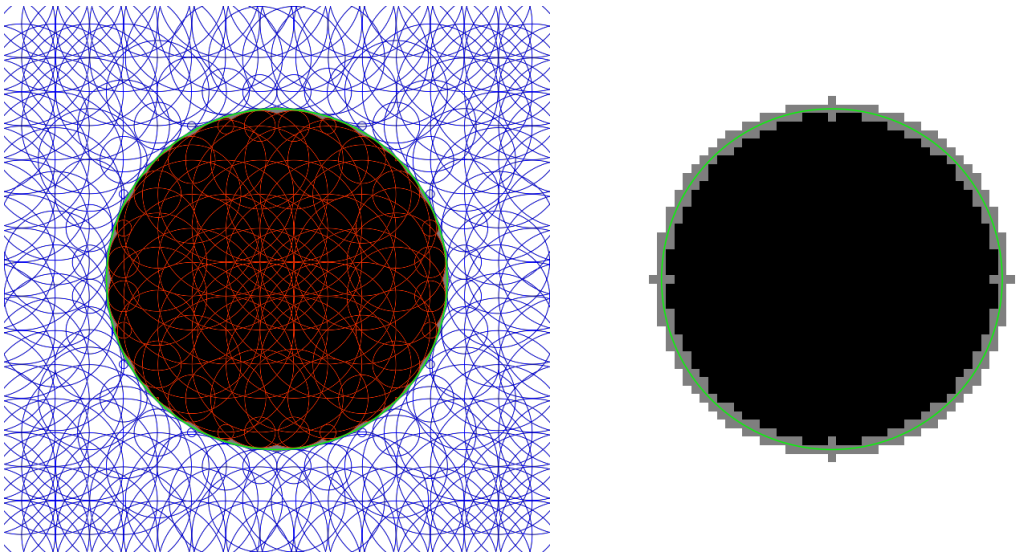
order of increasing $g(r)$, then identify the indices $0 = t_0 \le t_1 \le \cdots \le t_m$ such that $\mathcal{N}_u = \{ \mathcal{N}^*[j] : 0 \le j < t_u \}$, for each u.

Algorithm 15: `CDR-TO-TERNARY`

Input: A CDR $\mathbf{K} = (\mathtt{K}, \sigma, v, m, \gamma)$; a new cell size σ' that is a sub-multiple of σ
Output: A TIR $\mathbf{J} = (\mathtt{I}, \sigma', v')$, with $\text{int}(\mathbf{J}) \subseteq \text{int}(\mathbf{K})$, $\text{ext}(\mathbf{J}) \subseteq \text{ext}(\mathbf{K})$).

$k = \sigma/\sigma'$
$N = \mathcal{D}(\mathtt{K})$; $M = kN$
$\mathtt{J} =$ `NEW-IMG-ARRAY`$(M, \{-1, 0, +1\})$; Fill $\mathtt{J}$ with 0
Compute neighborhoods $\mathcal{N}_0$, $\mathcal{N}_1$, …, $\mathcal{N}_m$
for each $p \in \widehat{N}$ **do**
  $u = \mathtt{K}[p]$
  for each $r \in \mathcal{N}_u$ **do**
    $\mathtt{J}[kp + r] = \text{sgn}(u)$
return $(\mathtt{J}, \sigma', \text{sgn}(v))$

See Figure 8.1.



(a) Input (b) Output

Figure 8.1: Conversion of a clipped distance representation **K** (left) for a disk with radius 5 centered at $(8, 8)$ (green outline) to a ternary image representation (right), by Algorithm 15. The input CDR has cell size $\sigma = 1$, image size 16×16, $m = 127$, $\delta = 2\sigma = 2$ (so that $\gamma = 2/127$). The output TIR has cell size $\sigma = 1/4$ (refinement factor $k = 4$) and image size 64×64. All dimensions are in mm. The grid lines show pixel boundaries.

Chapter 9

Conversion from CDR to BMR

In this chapter we describe an algorithm to convert a clipped distance representation $\mathbf{K} = (\mathtt{K}, \sigma, v, m, \gamma)$ into a boundary mesh $\mathbf{B} = (\mathtt{B}, v')$.

Exact conversion is obviously impossible, since the interior and exterior of $\mathbf{K}$ are finite unions of open balls, while the boundary of $\mathbf{B}$ is piecewise flat. Ideally, we would like the conversion to be at least conservative, in the sense that

$$\mathrm{int}(\mathbf{B}) \subseteq \mathrm{int}(\mathbf{K}) \qquad \mathrm{ext}(\mathbf{B}) \subseteq \mathrm{ext}(\mathbf{K}) \qquad \partial\mathbf{B} \supseteq \partial\mathbf{K} \tag{9.1}$$

However, that is not possible either, because $\partial\mathbf{B}$ has zero thickness, while $\partial\mathbf{K}$ usually has non-empty interior. In theory, it is possible to construct $\mathbf{B}$ so that it satisfies the symmetrical constraints, namely

$$\mathrm{int}(\mathbf{K}) \subseteq \mathrm{int}(\mathbf{B}) \qquad \mathrm{ext}(\mathbf{K}) \subseteq \mathrm{ext}(\mathbf{B}) \qquad \partial\mathbf{K} \supseteq \partial\mathbf{B} \tag{9.2}$$

That is, $\mathbf{K}$ would be a conservative representation of $\mathbf{B}$, rather than the other way around. However, the construction would be rather complex and would require exact computation with certain non-rational numbers. That complexity would be hard to justify in most applications.

Therefore we will instead describe a conversion algorithm that is only approximate. The interior, exterior, and boundary of $\mathbf{B}$ will be generally close to those of $\mathbf{B}$, with errors on the order of the cell size σ. The quantification of those errors will be discussed in Chapter 10.

9.1 Overview of the algorithm

The algorithm is based on the *marching cubes* approach [28]. It covers the space $\mathbb{R}^n$ with an infinite collection of n-dimensional simplices with pairwise disjoint interiors, that constitute the *sampling mesh* $\mathbf{S}$. Then it assigns to each vertex q of $\mathbf{S}$ a *nominal distance value* $v(q)$, based on the CDR values $\mathbf{K}[p]$ at nearby voxels p. These values are implicitly extended to a continuous *approximate distance function* v from $\mathbb{R}^n$ to $\mathbb{R}$ by affine (barycentric) interpolation of corner values inside each simplex of $\mathbf{S}$ as in Equation (2.5). The output mesh $\mathtt{B}$ will be locus of all points $x \in \mathbb{R}^n$ such that $v(x) = 0$. To obtain

the mesh, the algorithm identifies, within each simplex of $\mathbf{S}$, the set P of points x with $v(x) = 0$. If not empty, the set P is decomposed into $(n-1)$-simplices that are added to `B`.

The sampling mesh. In this dissertation, we obtain the mesh $\mathbf{S}$ by considering every hypercube of side σ whose corners are centers of adjacent voxels of $\mathbf{S}$. Each of these hypercubes R is split into simplices in a fixed way.

Nominal corner distance values. With this construction, every vertex q of $\mathbf{S}$ is the center of a voxel $\mathbf{K}[p]$ of the CDR, namely $p = q/\sigma - (1/2)\mathbf{1}^n$. Therefore, the algorithm sets its nominal value $v(q)$ to the signed distance implied by the CDR, namely $\gamma\mathbf{K}[p]$.

Main loop. In the inner loop, the algorithm scans every simplex t of $\mathbf{S}$. If v is strictly positive at every vertex q of B, then v will be strictly positive everywhere in t. Likewise, if v is strictly negative at every corner q, then v will be strictly negative everywhere in t.

On the other hand, if t has at least two corners q and q' with $v(q) < 0 < v(q')$, the equation $v(x) = 0$ defines a hyperplane H_t that cuts t and separates those two vertices. The intersection P_t of t at H_t must be a flat convex polytope of dimension $n-1$, The algorithm then decomposes P_t into one or more $(n-1)$-dimensional simplices which are added to `B`.

Since the interpolated function v is continuous over all $\mathbb{R}^n$, the simplices in `B` will share faces properly, no matter in which order the simplices of $\mathbf{S}$ are scanned.

Degenerate cases. A complication occurs if the function v is zero at one or more corners of t. If there are at most $n-1$ null corners, the algorithm works, but the resulting mesh $\mathbf{B}$ may not be a manifold: it may have simplices that share faces in non-manifold ways. If exactly n corners are null, that entire face of t will be added to `B`, *twice*.

If all $n+1$ corners of t are null, then v will be zero over the whole simplex t. Then the set of all x such that $v(x) = 0$ will include t, and will not be an $(n-1)$-dimensional manifold.

These degenerate cases can be avoided by perturbing the nominal distance values $v(q)$ so that none of them is zero.

9.2 Algorithm

The above ideas are formalized in Algorithm 16. This algorithm assumes that exist a procedure called `SIMPLEX-COVER` that receives a CDR $\mathbf{K}$ and returns a set of n-simplices that are the sampling mesh $\mathscr{S}$ of the domain of $\mathbf{K}$. We briefly discuss this problem and give a solution in Appendix A.

The `INTERSECT-BOUNDARY` procedure returns **`true`** if the simplex t intersects the assumed boundary of $\mathbf{K}$ as described above; namely, if the values assigned to the corners of t do not have all the same sign. In that case, the algorithm calls the procedure `FIND-CROSSING-POINTS` to compute the set C of points where the edges of t intersect the

hyperplane H_t. Finally, DECOMP-HULL will construct the convex hull of C, and decompose it into $(n-1)$-simplices, which are added to $\mathbf{B}$.

Algorithm 16: CDR-TO-BOUNDARY

Input: A CDR $\mathbf{K} = (\mathtt{K}, \sigma, v, m, \gamma)$.

Output: A boundary representation $\mathbf{B} = (B, v')$ that approximately separates $\mathrm{int}(K)$ from $\mathrm{ext}(K)$

```
1 B = {}
2 v' = sgn(v)
3 𝒮 = SIMPLEX-COVER(K) for each simplex t ∈ 𝒮 do
4 |   if INTERSECT-BOUNDARY(K, t) then
5 |   |   C = FIND-CROSSING-POINTS(K, t)
6 |   |   L = DECOMP-HULL(C)
7 |   |   B = B ∪ L
8 return B(B, v')
```

Part IV

Experiments

Chapter 10

Accuracy of boundary extraction

The accuracy of shape representation is very important for CAD/CAM applications. In this chapter, we will analyze empirically the accuracy of the CDR and the errors introduced by conversion from it to the BMR.

10.1 Method

Figure 10.1 is a diagram of one iteration of the experiments. The chosen test shape B was a ball radius 1 centered at $(2, 2)$, all in mm. The IDR of B (interval function `PROCEDURAL-BALL`, Algorithm 5) was converted to the BIR and to the CDR with a specified resolution $1/\sigma$ (mm^{-1}), using Algorithms 8 and 12, respectively. Triangular meshes approximating the boundary surface of B were then extracted from these representations using Algorithms 10 and 16, respectively. Finally, these triangle meshes were compared to the ideal surface of B (the sphere of radius 1 mm centered at the origin) according to various error metrics.

This process was repeated several times with the same shape B, varying the resolution $1/\sigma$ from 2 to 512 mm^{-1}, doubling the value of $1/\sigma$ in each iteration. Note that this approach is equivalent to using a constant voxel size $\sigma = 1$ mm and varying the ball radius from 2 to 128 mm; and then expressing the errors relative to the ball radius, instead of absolute in mm. We also repeated the some of these experiments with different values of the quantization parameter m.

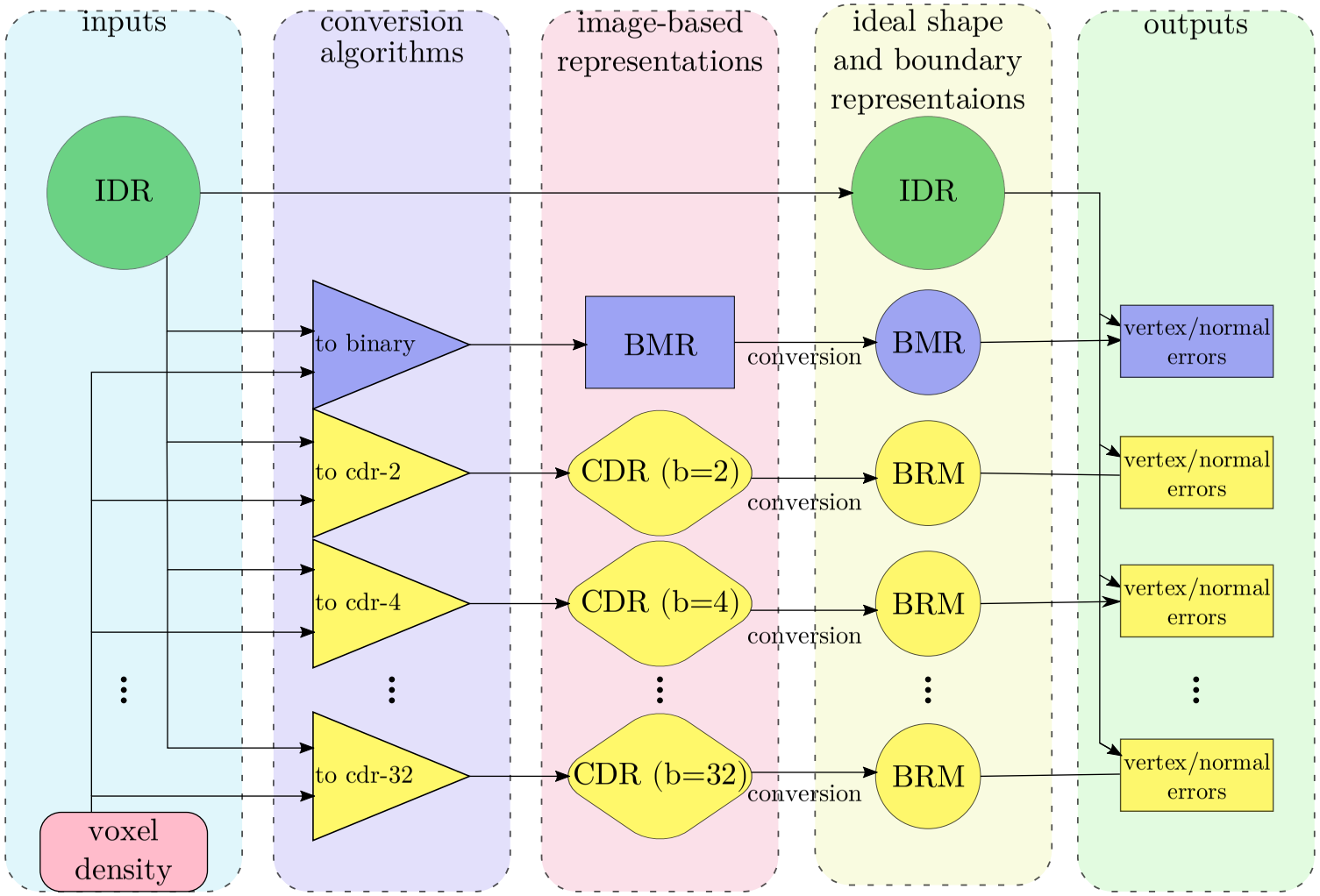


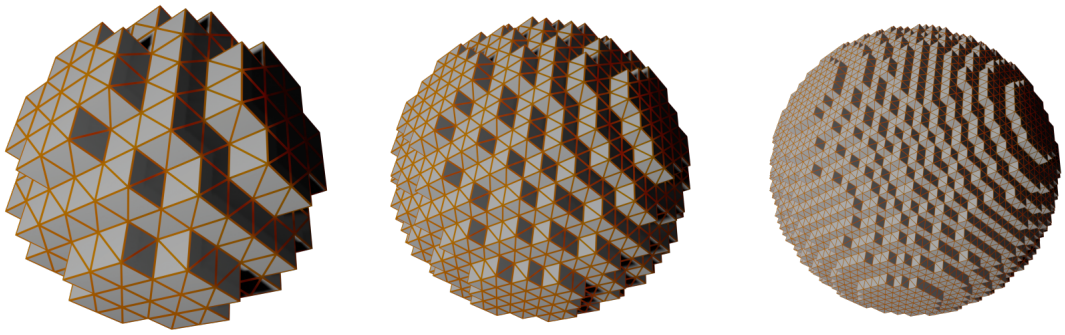
Figure 10.1: Diagram of one iteration of the experiments.

10.1.1 Image representations

Since voxels whose center is at more than $\sigma\sqrt{3}$ from the boundary are not used by Algorithm 16, we set $\delta = \sigma\sqrt{3}$ in all experiments. Both image-based representations were generated with sufficient domain size to accommodate the ball B plus a margin of $\sigma\lceil\sqrt{3}\rceil = 2\sigma$ (mm) all around. Therefore, the volume of the domain box $\mathcal{B}$ of both images was $V = (2 + 4\sigma)^3$ (mm^3) and the number of voxels was $(2\lceil 1/\sigma\rceil + 2)^3 \approx 8/\sigma^3$.

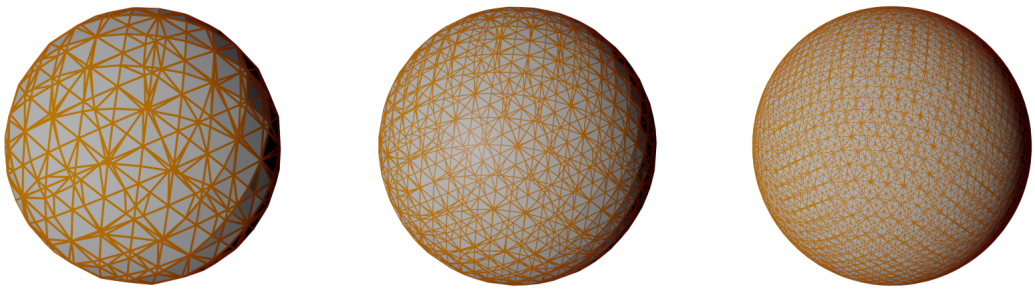
10.1.2 Extracted boundary meshes

Figures 10.2 and 10.3 show the boundary meshes obtained from the binary and CDR models of the ball B for different resolutions $1/\sigma$.



(a) $1/\sigma = 4\,\mathrm{mm}^{-1}$ (b) $1/\sigma = 8\,\mathrm{mm}^{-1}$ (c) $1/\sigma = 16\,\mathrm{mm}^{-1}$

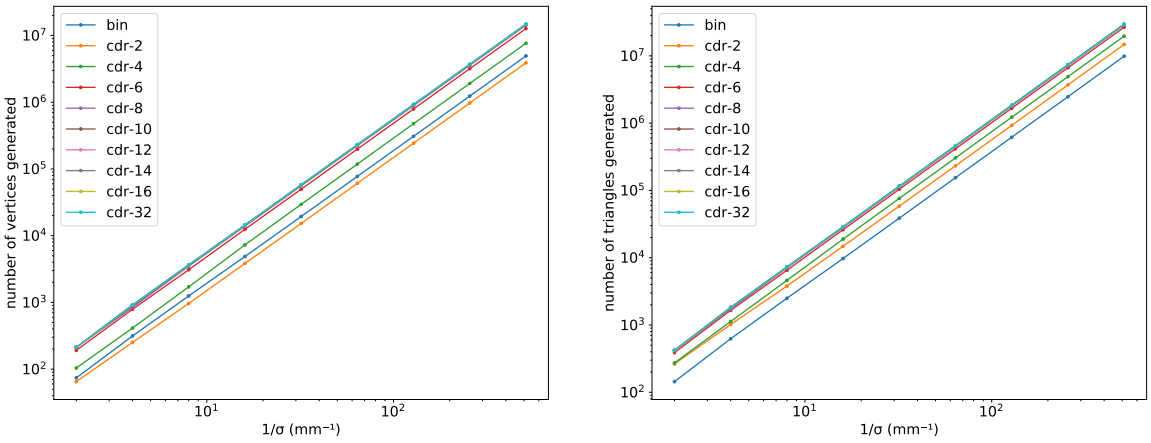
Figure 10.2: Boundary representations extracted from BIRs of the test ball.



(a) $1/\sigma = 4\,\mathrm{mm}^{-1}$ (b) $1/\sigma = 8\,\mathrm{mm}^{-1}$ (c) $1/\sigma = 16\,\mathrm{mm}^{-1}$

Figure 10.3: Boundary representations extracted from CDRs of the test ball with $m = 127$ ($b = 8$).

Figures 10.4a and 10.4b show the counts of vertices N_V and of triangles N_T, respectively, in the extracted BMRs. Note that, as expected, they both grow proportionally to $1/\sigma^2$.



(a) (b)

Figure 10.4: Variation of vertex count N_V (left) and triangle count N_T (right) as a function of the image resolution $1/\sigma$, for the triangle meshes obtained from the BIR and from the CDR with quantization parameter $m = 127$ ($b = 8$ bits/voxel).

10.1.3 Error metrics

We computed two different measures of representation error: *vertex position error* and *normal error*. For each metric $\phi = (\phi_0, \phi_1, \ldots, \phi_{N-1})$, we computed the weighted root-mean-square average

$$\mathrm{RMS}(\phi) = \sqrt{\frac{\sum_{i=0}^{N-1} w_i \phi_i^2}{\sum_{i=0}^{N-1} w_i}} \tag{10.1}$$

where w_i is the approximate area of the surface element sampled by ϕ_i.

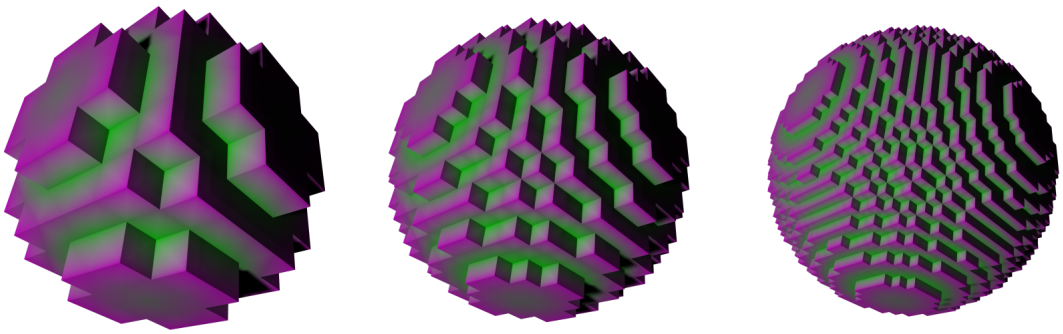
10.1.4 Storage size

For each experiment we also considered the amount of storage that would be required to store the corresponding image representation (binary or CDR) using a compact image format, such as run-length encoding [24], an n-dimensional binary tree (*k-d tree*) [7], or an adaptively sampled array [15].

For all these data structures, the storage required is roughly proportional to the voxels that near to the boundary of the object. Specifically, for the BIR, we assumed that the total number of bits needed to store the image is equal to the number of voxels that are adjacent to voxels of the opposite value. For a CDR with distance quantization parameter m, we assumed the total number of bits needed would be equal to the number of *non-trivial* voxels, whose value is neither $+m$ nor $-m$, times the number of bits needed to store a voxel value, $b = \lceil \log_2(m+1) \rceil$.

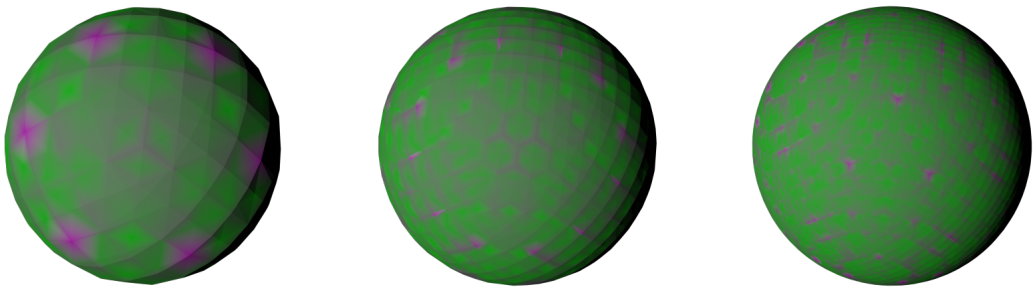
10.2 Vertex position error

The *vertex position error* at a vertex v_i of the mesh is the signed distance from v_i to the actual boundary of the ideal shape, namely $\phi_i = \|v_i\| - 1$ (mm) for the unit ball test shape B. See Figures 10.5 and 10.6.



(a) $1/\sigma = 4\,\text{mm}^{-1}$ (b) $1/\sigma = 8\,\text{mm}^{-1}$ (c) $1/\sigma = 16\,\text{mm}^{-1}$

Figure 10.5: Boundary representations extracted from BIRs, of the test ball B, as in Figure 10.2, with vertex colors based on the distance to the real boundary of the object. The color scale ranges from gray to magenta for points outside B, and from gray to green for vertices inside B; with the maximum saturation corresponding to the maxp and maxn (See Table 10.1) errors of each mesh, respectively.



(a) $1/\sigma = 4\,\text{mm}^{-1}$ (b) $1/\sigma = 8\,\text{mm}^{-1}$ (c) $1/\sigma = 16\,\text{mm}^{-1}$

Figure 10.6: Boundary representations extracted from CDRs of the test ball, as in Figure 10.3. Vertex colors are assigned based on the distance to the real boundary of the object, with the same color conventions as Figure 10.5 scaled to the maxp and maxn errors of each mesh.

We computed the vertex position errors with double precision floating point arithmetic. For this metric, the number N is the number N_{V} of vertices of the mesh, and the weight w_i is the sum of the areas of the triangles of the mesh that are incident to vertex v_i.

For this metric, besides the RMS average rmse, we also computed the maximum absolute value among the positive and negative errors ϕ_i, respectively maxp and maxn; as well as the weighted arithmetic average

$$\mathsf{avge} = \frac{\sum_{i=0}^{N-1} w_i \phi_i}{\sum_{i=0}^{N-1} \phi_i}. \tag{10.2}$$

The latter quantity could reveal whether the computed boundary is biased towards the convex or concave side of the surface. See Table 10.1 for some values. These error metrics are plotted in Figures 10.7 – 10.8 as a function of the resolution $1/\sigma$.

The CDRs with b from 2 to 12 mm has greater difference in rmse than those with the numbers of bits from 12 to 32 mm. For the rmse metric, in a ball, it seems that while m gets greater than $m > 8191$ ($b > 12$) the extra bits becomes less effective on the rmse metric.

In Figure 10.9, we plot this error as a function of the number of bits needed to store the non-trivials bits. In the plot we can see that from 157 bytes to 1 kB the CDR with $b = 8$ has smallest error; from 1 kB to 10 kB the smallest error is when $b = 10$; and for when $b = 12$, has the smallest error between the different CDR tested in the range 10 kB to 100 kB. Finally, for storage greater than 100 kB, the $b = 14$ and $b = 16$ seems not to make a significant difference in our tests for rmse.

It seems from the observation of these results, that there is a square ratio between the storage in bits and the error in this setup. That is, given the error constraint of $10^{-\epsilon}$ mm then we will use approximately $10^{\epsilon+2}$ bits.

Binary				
$1/\sigma$	maxn	maxp	avge	rmse
2	-0.29289	0.22474	0.02439	0.18590
4	-0.17084	0.17260	0.02380	0.10048
8	-0.08999	0.09687	0.00736	0.05213
16	-0.05214	0.05141	0.00246	0.02562
32	-0.02622	0.02603	0.00048	0.01282
64	-0.01315	0.01334	0.00036	0.00638
128	-0.00668	0.00672	-0.00001	0.00319

CDR $m = 7$ $(b = 4)$				
$1/\sigma$	maxn	maxp	avge	rmse
2	-0.15676	0.01661	-0.08536	0.09560
4	-0.08234	0.02452	-0.03329	0.04162
8	-0.04153	0.01352	-0.01514	0.01961
16	-0.02160	0.00721	-0.00740	0.00975
32	-0.01081	0.00360	-0.00362	0.00476
64	-0.00541	0.00180	-0.00182	0.00241
128	-0.00271	0.00090	-0.00090	0.00120

CDR $m = 31$ $(b = 6)$				
$1/\sigma$	maxn	maxp	avge	rmse
2	-0.09102	-0.00442	-0.03994	0.04464
4	-0.02853	0.00377	-0.01286	0.01432
8	-0.01050	0.00328	-0.00472	0.00540
16	-0.00520	0.00130	-0.00201	0.00236
32	-0.00257	0.00085	-0.00094	0.00113
64	-0.00129	0.00042	-0.00046	0.00055
128	-0.00064	0.00021	-0.00022	0.00027

CDR $m = 127$ $(b = 8)$				
$1/\sigma$	maxn	maxp	avge	rmse
2	-0.08491	0.00135	-0.02733	0.03351
4	-0.02064	0.00046	-0.00728	0.00863
8	-0.00624	0.00062	-0.00219	0.00256
16	-0.00216	0.00040	-0.00076	0.00086
32	-0.00075	0.00019	-0.00030	0.00034
64	-0.00032	0.00010	-0.00013	0.00015
128	-0.00016	0.00005	-0.00006	0.00007

Table 10.1: Maximum interior and exterior absolute vertex errors maxn and maxp, the signed weighted average vertex error avge, and the root-mean-square average vertex error rmse as a function of the image resolution $1/\sigma$ (mm), for the BIR and for the CDR with distance quantization parameter $m = 7$, 31, and 127 (that is, $b = 4$, 6, and 8 bits per voxel). All error values are in mm.

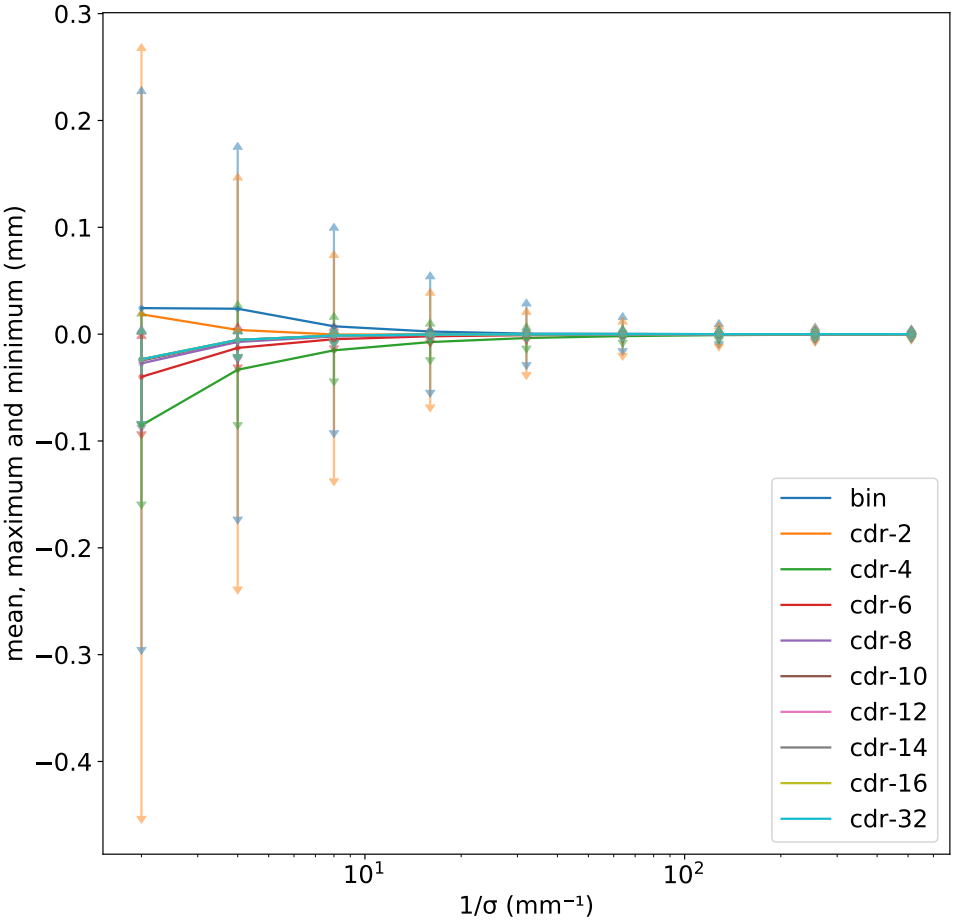


Figure 10.7: Maximum interior and exterior absolute vertex errors maxn, maxp and the signed weighted average error avge, as a function of the image resolution $1/\sigma$, for the distance quantization parameter $m = 127$ ($b = 8$).

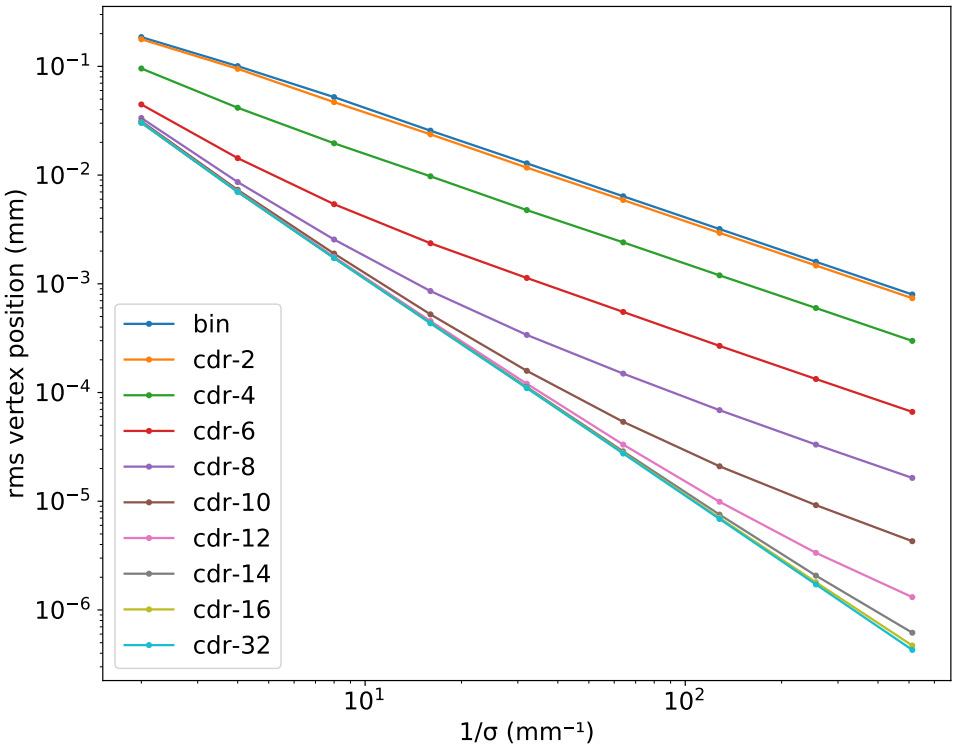


Figure 10.8: The RMS vertex error rmse of the boundary meshes obtained from the BIR and the CDR with the distance quantization parameter $m = 2^b/2 - 1$ where b is in $\{2, 4, 6, 8, 10, 12, 15, 16, 32\}$ and as a function of the image resolution $1/\sigma$.

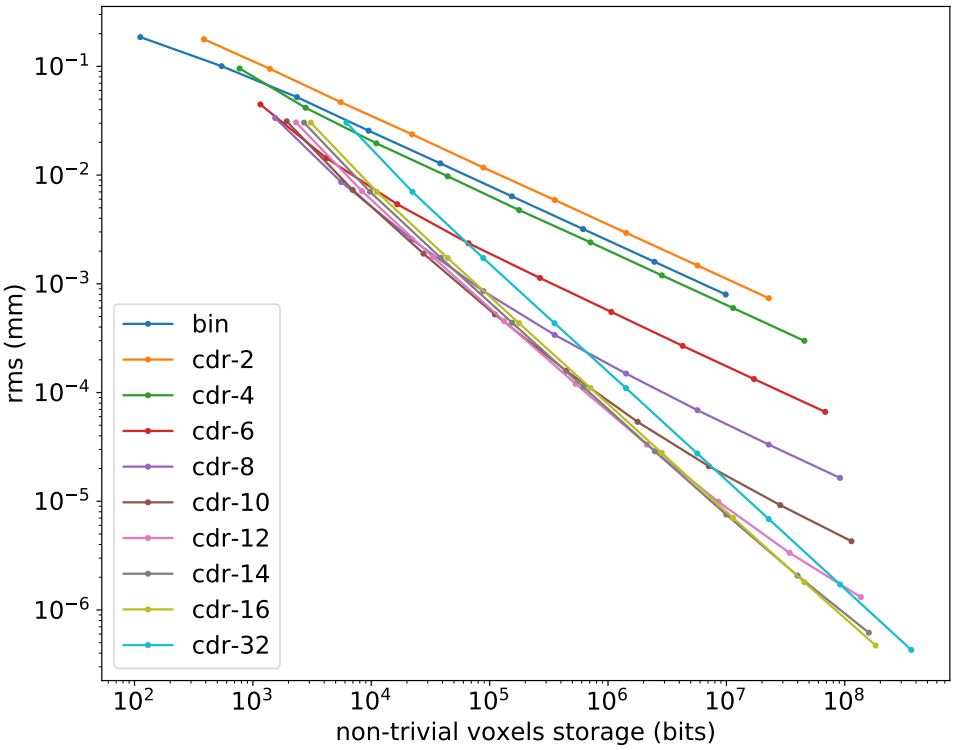
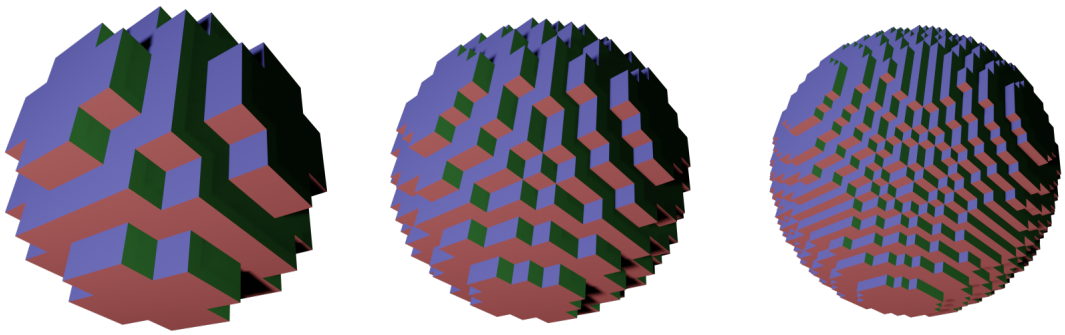


Figure 10.9: The RMS vertex error rmse, as a function of the estimated amount of memory (in bits) needed to store the non-trivial voxels, for the distance quantization parameter values $m = 2^b/2 - 1$ ($b = 2, 4, 8, 10, 12, 14, 16, 32$).

10.3 Normal error

The *normal* of a mesh triangle is the unit vector perpendicular to the triangle, directed towards the exterior side. See Figures 10.10 and 10.11.



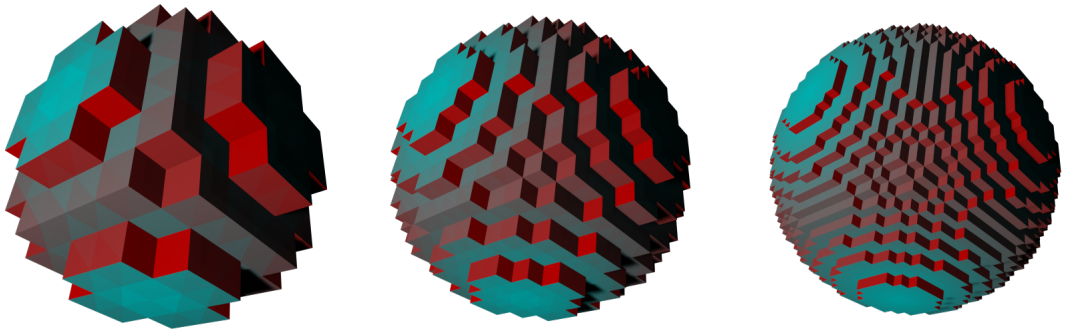
(a) $1/\sigma = 4\,\mathrm{mm}^{-1}$ (b) $1/\sigma = 8\,\mathrm{mm}^{-1}$ (c) $1/\sigma = 16\,\mathrm{mm}^{-1}$

Figure 10.10: Boundary representations extracted from BIRs of the test ball, as in Figure 10.2. Each triangle is colored according to the direction of its normal vector, with the three coordinates mapped to red, green, and blue intensities.

(a) $1/\sigma = 4\,\mathrm{mm}^{-1}$ (b) $1/\sigma = 8\,\mathrm{mm}^{-1}$ (c) $1/\sigma = 16\,\mathrm{mm}^{-1}$

Figure 10.11: Boundary representations extracted from CDRs of the test ball, as in Figure 10.3 Each triangle is colored according to the direction of its normal, as in Figure 10.10.

The *normal error* of a triangle of the mesh is the angle ϕ_i between its normal v_i and the normal of the ideal shape B at the point that is the projection of the triangle's centroid c_i onto the boundary of the sphere. Namely, $\phi_i = \arccos(v_i \cdot c_i / \|c_i\|)$ where '$\cdot$' is the inner product of $\mathbb{R}^3$. See Figures 10.12 and 10.13.

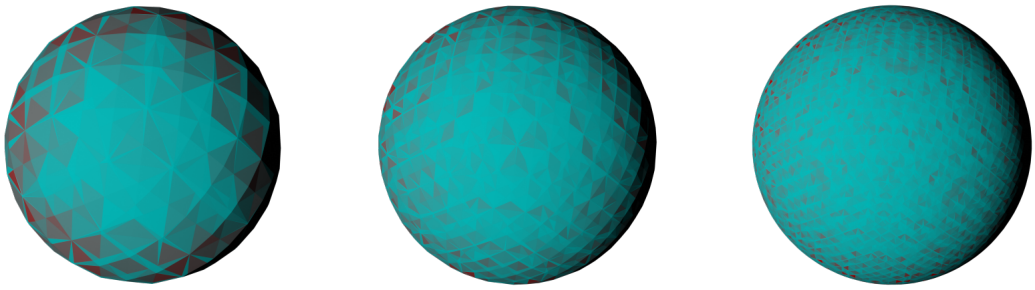


(a) $1/\sigma = 4\,\mathrm{mm}^{-1}$
$\mu = 0.117$; $\alpha = 1.327$

(b) $1/\sigma = 8\,\mathrm{mm}^{-1}$
$\mu = 0.0589$; $\alpha = 1.449$

(c) $1/\sigma = 16\,\mathrm{mm}^{-1}$
$\mu = 0.0295$; $\alpha = 1.509$

Figure 10.12: Boundary representations extracted from BIRs of the test ball, as in Figure 10.2, with each triangle colored according to the normal error. The minimum error (μ) is mapped to cyan, and the maximum error (α) is mapped to red.



(a) $1/\sigma = 4\,\mathrm{mm}^{-1}$
$\mu = 0$; $\alpha = 0.213$

(b) $1/\sigma = 8\,\mathrm{mm}^{-1}$
$\mu = 0$; $\alpha = 0.128$

(c) $1/\sigma = 16\,\mathrm{mm}^{-1}$
$\mu = 0$; $\alpha = 0.076$

Figure 10.13: Boundary representations extracted from CDRs of the test ball, as in Figure 10.3. Each triangle is colored according to its normal error, as in Figure 10.12.

For this metric, the number of measurements N is the number of triangles N_T, and the weight w_i of ϕ_i is the area of the triangle. The RMS average rmsn of this error is given in Table 10.2 and plotted in Figure 10.14.

Figure 10.14 shows that each quantization precision stabilizes while the resolution $1/\sigma$ increases.

In Figure 10.15, we plot the rmsn error as a function of the number of bits needed to store the non-trivial bits of the representations.

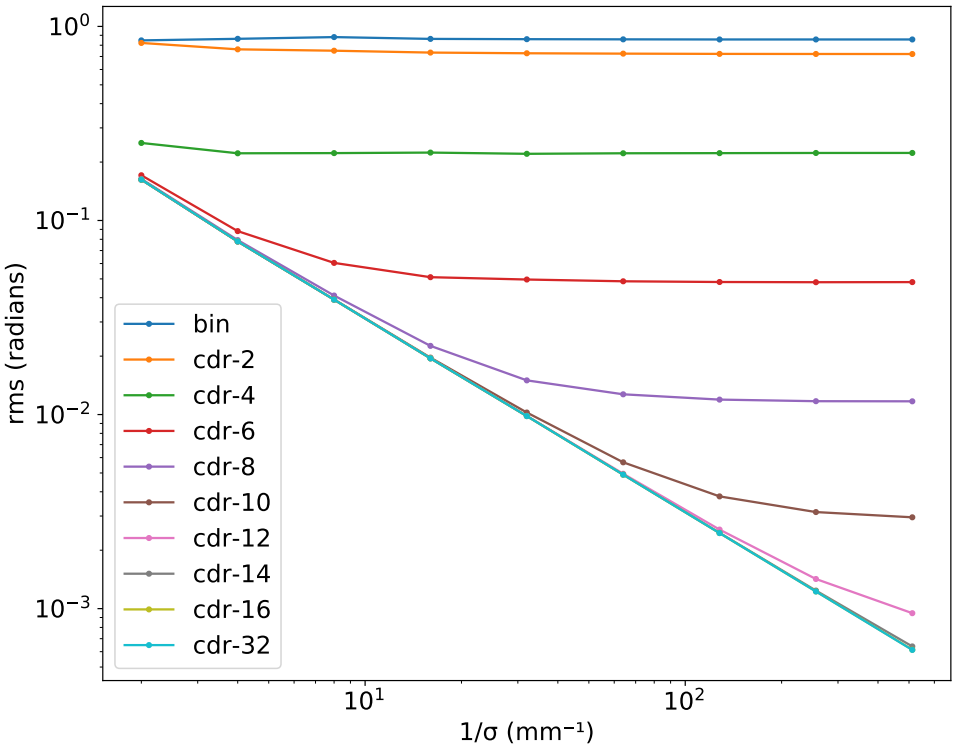


Figure 10.14: The RMS normal error as a function of the resolution $1/\sigma$, for the boundary meshes obtained from the BIR and from the CDR with quantization parameters $m = 2^b/2 - 1$ ($b = 2$, 4, 6, 8, 10, 12, 14, 16, and 32).

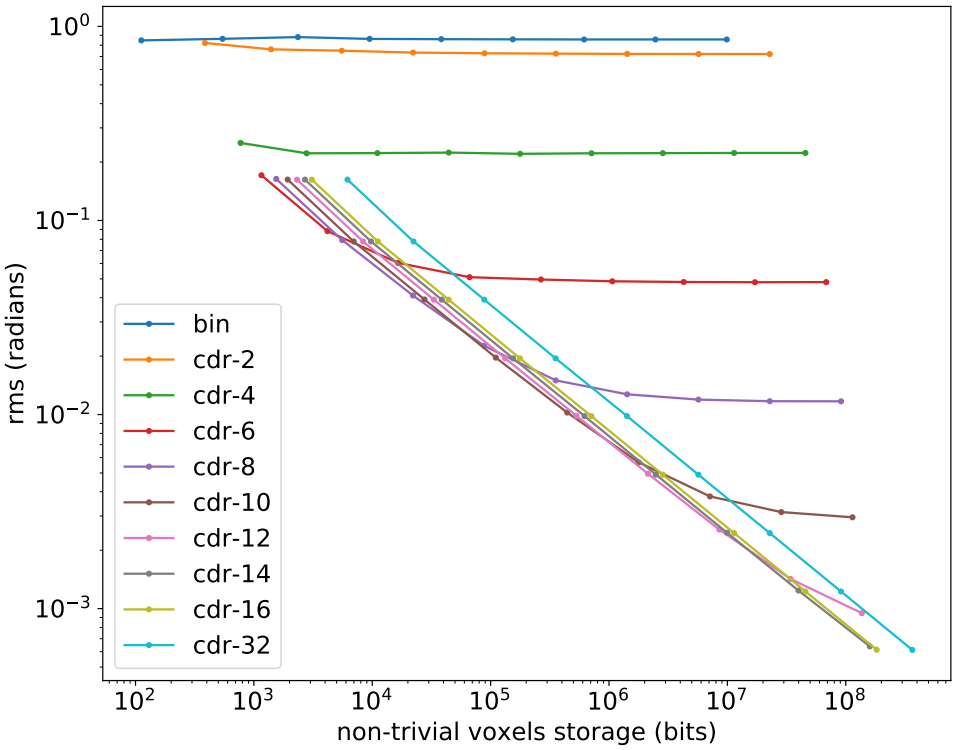


Figure 10.15: The RMS normal error of the boundary meshes obtained from the BIR and CDR representation with quantization parameters $m = 2^b/2 - 1$ ($b = 2$, 4, 6, 8, 10, 12, 14, 16 and 32 bits per voxel), as a function of the number of bits needed to store the compacted image arrays.

Binary	
$1/\sigma$	rmsn
2	0.84723
4	0.86317
8	0.88106
16	0.86237
32	0.85948
64	0.85779
128	0.85654

CDR $m = 7$ ($b = 4$)	
$1/\sigma$	rmsn
2	0.25093
4	0.22197
8	0.22244
16	0.22382
32	0.22058
64	0.22199
128	0.22234

CDR $m = 31$ ($b = 6$)	
$1/\sigma$	rmsn
2	0.17103
4	0.08815
8	0.06046
16	0.05103
32	0.04964
64	0.04861
128	0.04819

CDR $m = 127$ ($b = 8$)	
$1/\sigma$	rmsn
2	0.16378
4	0.07923
8	0.04105
16	0.02259
32	0.01500
64	0.01272
128	0.01194

Table 10.2: Root-mean-square average normal error rmsn as a function of the image resolution $1/\sigma$ (mm), for the BIR and for the CDR with distance quantization parameter $m = 7$, 31, and 127 (that is, $b = 4$, 6, and 8 bits per voxel). All error values are in mm.

Figure 10.15 shows that from 157 bytes to 1 kB the CDR with $b = 8$ has smaller error between the tested representations. From the range 250 bytes to approximately 2.4 kB, the smallest error is from the CDR with $b = 8$. The CDR with $b = 10$ has the smallest error in the range of 2.4 kB to 100 kB; and the smallest error of the range 100 kB to 1 MB is when $b = 12$. Finally, for storage size greater than 100 kB the CDRs with $b = 14, 16$ have similar errors.

For the configuration of this experiment, it seems that there is a exponential ratio between the storage in bits and the normal error. That is, given the error constraint of 10^{-e}, then we will use approximately $10^{2(1+\epsilon)}$.

Chapter 11

Implementation

In this section, we will describe the computer resources and the software we used to write this dissertation and realize the experiments.

While the examples in this dissertation were only 2D and 3D shapes, the main code base of this dissertation handles shapes of any dimension n.

The whole software is a package available on the Gitlab and is called `shrep`, for "shape representation" [29]. Almost all of it is coded in the C++17 programming language. The code was developed with the Unix philosophy in mind, so as to facilitate the creation of new programs, scripts, experiments.

The package comprises a library also called `shrep`, with procedures that implement the algorithms described in this dissertation and other basic functions like file input and output; and a suite of executable programs that call those algorithms to perform specific tasks such as generation of sample shapes, representation conversions, visualization, and analysis.

The `shrep` package depends on several other available GNU/Linux packages and libraries, including xtensor (version 0.21.4) [17], Opencv (4.2.0-2) [10], OpenGL, and OpenGL Mathematics (GLM) (0.9.9.7) [16]. The code was compiled with the Gnu C/C++ Compiler (9.2.1) using the CMake (3.16.4) build generator. We also used MeshLab (2016.12) [31] for visualization and debugging; and Blender 2.81 [13] to render the 3D representations of Figures 10.2, 10.3, 10.5, 10.6 and 10.10 – 10.13.

11.1 Data file formats

The package also defines a family of file formats to store the shape representations described in this dissertation. The files formats have the extensions `.shr.bin` (binary image representation), `.shr.ter` (ternary image) `.shr.cdr` (clipped signed distance), and `.shr.bd` (boundary mesh). The file formats are special cases of *JSON* (JavaScript Object Notation) files with some metadata like the representation name, the cell density of the representation, a domain offset, and others.

11.2 Package components

- `shrep` is the main library, that defines data structures for the various representations (BIR, TIR, CDR, BMR) as well as the procedures to manipulate them, including all the conversion algorithms described in the text.

  The library also includes the implementation of common interval arithmetics operations, based on the work of Stolfi and De Figueiredo [45], and of the IA functions described in Section 2.5.

- `shrgen` is an executable program that generates BIR, TIR, and CDR shape representations for some objects, from their procedural representations. It uses the conversion algorithms from the `shrep` library.

- `shrcon` is an executable program used to convert shapes between various representations. It too uses the conversion algorithms from the `shrep` library.

- `shrfile` shrfile is a program that converts the `.shr.*` file formats to other established formats. In particular, it converts 2D `.shr.bin` and `.shr.cdr` files to `.png` or `.jpeg` images; and 3D `.shr.bd` fles to `.obj` and `.ply` files.

- `shrview` is a interactive visualization tool. This program was used to generate the images of 2D shape representations (BIR, TIR, CDR, and BMR) used in this dissertation. Also, the figures of the union-of-balls interpretation of 2D CDR.

- `errcomp` is an executable program that generates the data, and shapes for the plots and rendering of Chapter 10.

Part V

Conclusion

Chapter 12

Conclusions

In this chapter, we will briefly discuss the results and contributions of this dissertation. Then, present the topics to extend this research project.

12.1 Results and contributions

In this dissertation, we presented a framework of shape representations and their topology semantics. We also describe conversion algorithm for shape representation that takes in consideration these semantics.

We focused, however, in the (discrete) clipped signed Euclidean distance representation (CDR), an image-based shape representation. Specifically, we defined mathematical properties about the CDR, the correctness and tightness properties. Also, we present an approach to reason about the CDR, the union-of-balls interpretation.

For the CDR and the BIR, we made experimental tests using the vertex position error and the normal error. For the test configuration and the metrics used, we observed a significant difference when choosing the quantization parameter $m = 2^b/2 - 1$ of the CDR for $2 \leq b \leq 12$.

In our tests, for resolutions $1/\sigma < 512$, we also observed that the RMS of the vertex error has a squared ratio with the chosen number of bits and, similarly, the RMS of the normal error has a exponential ratio with the number of bits.

12.2 Future work

In this dissertation, we study the clipped distance representation using the Euclidean distance. For some applications that uses n-dimensional data, the Euclidean distance has some problems [1]. Therefore, exploring this representation using different distance functions and its impacts and semantics in the geometric operations and conversions.

Also, we want to apply the union-of-balls interpretation, correctness, and tightness properties in conversion algorithms, and other common geometrical operations like union, intersection, minus, and morphology.

In the experiments we compare the different precision of the CDR and the BIR. Currently, as far as we know, there is experimental comparison between boundary extraction

of the TIR and CDR that takes into consideration the semantics of these representations. Namely, such algorithm must extract the BMR $\mathbf{B_T}$ of a TIR $\mathbf{T}$ and ensures that $\partial\mathbf{B_T} \subseteq \partial\mathbf{T}$; and similarly a BMR $\mathbf{B}_K$ such that $\partial\mathbf{B} \subseteq \partial\mathbf{K}$. For these algorithms, other property to investigate is that such boundary $\mathbf{B}$ has maximal distance as possible from the interior and exterior of $\mathbf{K}$.

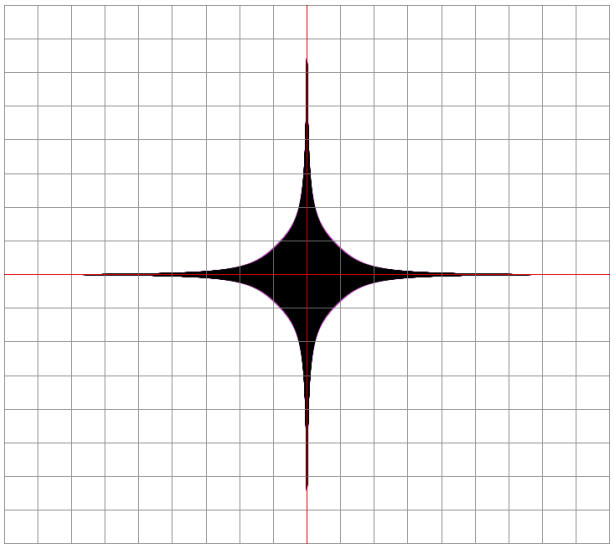


Figure 12.1: Object with sharp details.

Sphere usually is a good object to test representation techniques; but we can also have some insights with more complicated objects, like fractals, spirals, shells, and signed distance functions with sharp details. For instance, the signed distance function

$$f(x, y) = \min(\beta + \lambda, \beta - \lambda), \tag{12.1}$$

where $\beta = \sqrt{x^2 + y^2}$ and $\lambda = \frac{1}{xy}$. See Figure 12.1.

In the present day, CDR (or truncated distance field) is render using ray marching algorithm techniques, that has problems for construct normals and aliasing. The union of the balls model (Section 6.2) gives us a method to retrieve easily the normal of the balls from the model, and this normals are the real normals of the balls that composes the union. A render algorithm that uses the union of balls model can improve the quality of the render image.

Bibliography

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

[2] A. Baer, C. Eastman, and M. Henrion. Geometric modelling: a survey. *Computer-Aided Design*, 11(5):253–272, 1979.

[3] A. Bærentzen and N. J. Christensen. A technique for volumetric csg based on morphology. In *Volume Graphics 2001*, pages 117–130. Springer, 2001.

[4] J. A. Bærentzen and N. J. Christensen. Volume sculpting using the level-set method. In *Proceedings SMI. Shape Modeling International 2002*, pages 175–275. IEEE, 2002.

[5] J. A. Bærentzen, M. Šrámek, and N. J. Christensen. A morphological approach to the voxelization of solids. *Journal of the WSCG*, 8(1-3), 2000.

[6] M. Baker et al. Computer graphics, c version. 1997.

[7] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[8] A. Bliss and F. E. Su. Lower bounds for simplicial covers and triangulations of cubes. *Discrete & Computational Geometry*, 33(4):669–686, Apr 2005. ISSN 1432-0444. doi: 10.1007/s00454-004-1128-0. URL https://doi.org/10.1007/s00454-004-1128-0.

[9] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in cagd. *Computer Aided Geometric Design*, 1(1):1–60, 1984.

[10] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] D. R. Canelhas, E. Schaffernicht, T. Stoyanov, A. J. Lilienthal, and A. J. Davison. An eigenshapes approach to compressed signed distance fields and their utility in robot mapping. *arXiv preprint arXiv:1609.02462*, 2016.

[12] F. Chazal and A. Lieutier. Topology guaranteeing manifold reconstruction using distance function to noisy data. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 112–118. ACM, 2006.

[13] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

[14] M. Ferrant, B. Macq, A. Nabavi, and S. K. Warfield. Deformable modeling for characterizing biomedical shape changes. In *International Conference on Discrete Geometry for Computer Imagery*, pages 235–248. Springer, 2000.

[15] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 249–254, 2000.

[16] K. Group. Opengl mathematics. https://glm.g-truc.net/0.9.8/index.html, March 2020.

[17] X. S. Group. Xtensor - multi-dimensional arrays with broadcasting and lazy computing. https://github.com/xtensor-stack/xtensor, March 2020.

[18] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1432–1439, 2007.

[19] M. Haiman. A simple and relatively efficient triangulation of then-cube. *Discrete & Computational Geometry*, 6(3):287–289, Sep 1991. doi: 10.1007/BF02574690. URL https://doi.org/10.1007/BF02574690.

[20] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE transactions on pattern analysis and machine intelligence*, PAMI-9(4):532–550, 1987.

[21] T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. *Journal of the ACM (JACM)*, 48(5):1038–1068, 2001.

[22] R. B. Hughes and M. R. Anderson. Simplexity of the cube. *Discrete Mathematics*, 158(1-3):99–150, 1996.

[23] IEEE. Standard for floating-point arithmetic. Technical Report IEEE Std 754-2008, International Organization for Standardization, June 2008.

[24] M. W. Jones, J. A. Baerentzen, and M. Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on visualization and Computer Graphics*, 12(4):581–599, 2006.

[25] A. Kaufman. Volume visualization. *The visual computer*, 6(1):1–1, 1990.

[26] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *Computer*, 26(7):51–64, 1993.

[27] S. Lavallée and R. Szeliski. Recovering the position and orientation of free-form objects from image contours using 3d distance maps. *IEEE Transactions on pattern analysis and machine intelligence*, 17(4):378–390, 1995.

[28] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[29] G. Y. B. Matsuzake. shrep - shape representation library. https://gitlab.com/yudi-matsuzake/shrep, March 2020.

[30] D. J. Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory, 1980.

[31] Meshlab. Meshlab. http://www.meshlab.net/, March 2020.

[32] H. Minkowski. Volumen und oberfläche. In *Ausgewählte Arbeiten zur Zahlentheorie und zur Geometrie*, pages 146–192. Springer, 1989.

[33] R. E. Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.

[34] R. E. Moore. *Methods and applications of interval analysis*. SIAM, 1979.

[35] P. Novotny, L. I. Dimitrov, and M. Sramek. Enhanced voxelization and representation of objects with sharp details in truncated distance fields. *IEEE transactions on visualization and computer graphics*, 16(3):484–498, 2009.

[36] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart. Signed distance fields: A natural representation for both mapping and planning. In *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan, 2016.

[37] D. Orden and F. Santos. Asymptotically efficient triangulations of the d-cube. *Discrete & Computational Geometry*, 30(4):509–528, Oct 2003. ISSN 1432-0444. doi: 10.1007/s00454-003-2845-5. URL https://doi.org/10.1007/s00454-003-2845-5.

[38] W. Regli, J. Rossignac, V. Shapiro, and V. Srinivasan. The new frontiers in computational modeling of material structures. *Computer-Aided Design*, 77:73–85, 2016.

[39] U. Rembold and R. Dillmann. *Computer-Aided Design and Manufacturing*. Springer, 1984.

[40] A. G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys (CSUR)*, 12(4):437–464, 1980.

[41] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM (JACM)*, 13(4):471–494, 1966.

[42] SIGGRAPH 1974. *SIGGRAPH '74: Proceedings of the 1st Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1974. Association for Computing Machinery. ISBN 9781450373524.

[43] W. D. Smith. A lower bound for the simplexity of then-cube via hyperbolic volumes. *European Journal of Combinatorics*, 21(1):131 – 137, 2000. ISSN 0195-6698. doi: https://doi.org/10.1006/eujc.1999.0327. URL http://www.sciencedirect.com/science/article/pii/S019566989990327X.

[44] M. Sramek and A. E. Kaufman. Alias-free voxelization of geometric objects. *IEEE transactions on visualization and computer graphics*, 5(3):251–267, 1999.

[45] J. Stolfi and L. H. De Figueiredo. *Self-validated numerical methods and applications*. Number 5 in Monographs of the 21st Brazilian Mathematics Colloquium. IMPA, Rio de Janeiro, 1997.

[46] N. Volpato, A. L. J. Munhoz, C. A. Costa, C. H. Ahrens, J. de Carvalho, J. R. L. dos Santos, J. V. L. da Silva, J. A. Foggiatto, and M. S. F. de Lima. *Manufatura Aditiva: Tecnologias e aplicações da impressão 3D*. Blucher, 2017.

[47] H. Zhu, Y. Liu, J. Bai, and X. Ye. Constructive generation of the medial axis for solid models. *Computer-Aided Design*, 62:98–111, 2015.

Part VI

Appendices

Appendix A

Decomposing a hypercube into simplices

In this chapter we will discuss the problem of decomposing an n-dimensional hypercube $\mathbf{K} = [0,1]^n = [0,1] \times [0,1] \times \cdots \times [0,1]$ into n-dimensional simplices. This step is required by the method that we described for conversion of CDR to BMR (Algorithm 16).

Let $\mathcal{V} = \{v_0, v_1, \ldots, v_{m-1}\}$ be the list of the vertices of the hypercube, in lexicographic order of their coordinates. There are many ways to perform that subdivision, even if we require that the vertices of each simplex be a subset of $\mathcal{V}$. (Such a decomposition is called a *simplexity* by some authors.) For instance, a cube ($n = 3$) can be divided into six tetrahedra in 12 different ways, and into 5 tetrahedra in just one way. The smallest number of simplices known for $n = 2, 3, 4, 5, 6$ and 7 is respectively $2, 5, 16, 67, 308$ and 1493 [37].

The following method is simple to implement. Let $\mathcal{P}$ be the set of all permutations of $\{0, \ldots, n\}$. For each permutation $\pi \in \mathcal{P}$, let S_π be the simplex

$$S_\pi = \left\{ (x_0, \ldots, x_n) \in \mathbb{R}^n : 0 \le x_{\pi(0)} \le x_{\pi(1)} \le \cdots \le x_{\pi(n-1)} \le 1 \right\}. \tag{A.1}$$

Then the set

$$\mathscr{S} = \{ r(S_\pi + c) : \pi \in \mathcal{P} \}, \tag{A.2}$$

is a decomposition of $\mathbf{K}$.

This decomposition is optimal for $n = 2$, but not optimal for $n \ge 3$. For instance, for $n = 3$ it gives $3! = 6$ tetrahedra, instead of the optimal 5. For $n = 4$ it gives 24 4-simplices, instead of the optimal 16.

Finding the simplexity with minimal cardinality is still an open problem. Even the size of that simplexity is not known for general n, but some lower bounds and upper bounds are known [8, 19, 22, 37]. In the work of Smith [43] is found the lower bound of

$$\frac{1}{2}\left(6^{n/2}(n+1)^{-\frac{n+1}{n}} n!\right)$$

and Orden and Santos [37] found an algorithm that build a simplexity with cardinality in $O(0.816^n n!)$.

Appendix B

The largest ball problem

In this chapter we present a theoretical algorithm to solve the *largest included ball* problem — a computational geometry problem that would have to be solved in order to produce a tight (maximally explicit) CDR, as discussed in Section 6.4.

B.1 Statement of the problem

Let $\mathscr{B}$ be a finite set of n-dimensional closed balls with pairwise distinct centers, and p be a point in the set $\bigcup\mathscr{B}$. It is required to find the largest ball centered at p that is contained in $\bigcup\mathscr{B}$.

B.2 Theory

Lemma 1. *The radius r of the largest ball centered at p that is contained in $\bigcup\mathscr{B}$ is the distance from p to the boundary of $\bigcup\mathscr{B}$.*

The Lemma 1 is obvious, therefore finding the largest ball problem reduces to computing the distance $d(p, \partial(\bigcup\mathscr{B}))$.

To compute this distance, we will partition the boundary of $\bigcup\mathscr{B}$ into a finite number of objects that are simpler than the boundary of $\bigcup\mathscr{B}$. For this purpose, we will define some objects that will help us to reason about this problem.

J-sphere. Given a nonempty subset $\mathscr{J} \subseteq \mathscr{B}$ with the cardinality $|\mathscr{J}|$ at most n, we define the *$\mathscr{J}$-sphere* $\mathcal{S}(\mathscr{J})$ as

$$\mathcal{S}(\mathscr{J}) = \bigcap_{B\in\mathscr{J}} \partial(B). \tag{B.1}$$

Note that the object $\mathcal{S}(\mathscr{J})$ may be empty, a point or a $(n-1)$-dimensional hypersphere in a n-dimensional space, where $1 \leq k \leq n$.

J-piece. We also define the *$\mathscr{J}$-piece* of $\partial(\bigcup \mathscr{B})$ as

$$\mathcal{P}(\mathscr{B}, \mathscr{J}) = \mathcal{S}(\mathscr{J}) \cap \partial\left(\bigcup \mathscr{B}\right). \tag{B.2}$$

Candidates objects. The set of *candidate objects* with dimension z or less in the boundary of $\bigcup \mathscr{B}$ is

$$\mathcal{C}'(\mathscr{B}, z) = \{\mathcal{P}(\mathscr{B}, \mathscr{J}) : \mathscr{J} \subseteq \mathscr{B} \wedge |\mathscr{J}| = n - z\}, \tag{B.3}$$

where $0 \leq z < n$. Finally, the set of candidate objects with exact dimension z is

$$\mathcal{C}(\mathscr{B}, z) = \begin{cases} \mathcal{C}'(\mathscr{B}, z) & \text{if z} = 0 \\ \mathcal{C}'(\mathscr{B}, z) \setminus \mathcal{C}'(\mathscr{B}, z-1) & \text{if } 0 < \text{z} < \text{n}. \end{cases} \tag{B.4}$$

For instance, given the set $X = \{A, B\}$ where A and B are 2d closed balls, $\mathcal{P}(X, X)$ can be empty if A and B does not intersect, a point if the boundary of A only touches the boundary of B in one point and otherwise a 1d sphere in a 2d space. Note that the 1-dimensional ball is an interval, and the boundary of the interval (1-dimensional sphere) is two points. See Figure B.1.

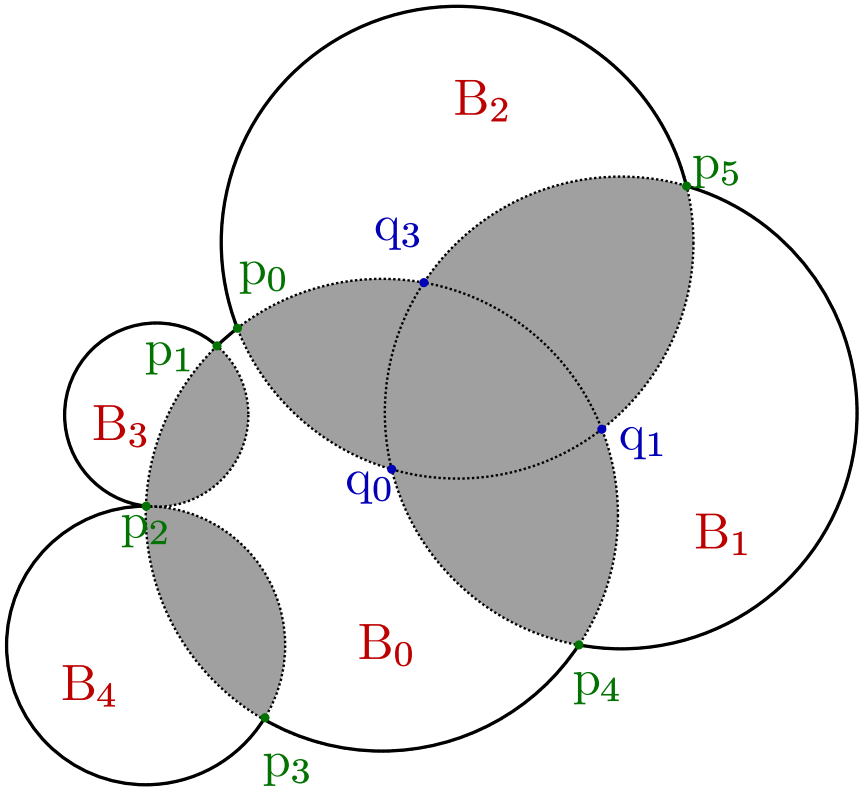


Figure B.1: Example of a set $\mathscr{B} = \{B_0, \ldots, B_4\}$ of 2d balls, where the balls boundary is the dotted curves, the boundary of $\bigcup \mathscr{B}$ is solid curves, and $\mathcal{C}'(\mathscr{B}, 0)$ is the set $\{\{p_0\}, \{p_1, p_2\}, \{p_2, p_3\}, \{p_4\}, \{p_5\}\}$.

Lemma 2. *Given a set $\mathscr{B}$ of n-dimensional closed balls with pairwise distinct centers,* $\bigcup \mathcal{C}'(\mathscr{B}, n-1) = \partial(\bigcup \mathscr{B})$

Proof. By definition we know that

$$\begin{aligned}\mathcal{C}'(\mathscr{B}, n-1) &= \{\mathcal{P}(\mathscr{B}, \mathscr{J}) : \mathscr{J} \subseteq \mathscr{B} \wedge |\mathscr{J}| = 1\} \\ &= \{\mathcal{P}(\mathscr{B}, \{B\}) : B \in \mathscr{B}\} \\ &= \left\{\mathcal{S}(\{B\}) \cap \partial\left(\bigcup \mathscr{B}\right) : B \in \mathscr{B}\right\} \\ &= \left\{\partial(B) \cap \partial\left(\bigcup \mathscr{B}\right) : B \in \mathscr{B}\right\},\end{aligned}$$

then

$$\bigcup \mathcal{C}'(\mathscr{B}, n-1) = \bigcup_{B \in \mathscr{B}} \left(\partial(B) \cap \partial\left(\bigcup \mathscr{B}\right)\right),$$

and by the distributive property of the intersection operation over union we have

$$\bigcup \mathcal{C}'(\mathscr{B}, n-1) = \left(\bigcup_{B \in \mathscr{B}} \partial(B)\right) \cap \partial\left(\bigcup \mathscr{B}\right).$$

Finally, we know that

$$\partial\left(\bigcup \mathscr{B}\right) \subseteq \bigcup_{B \in \mathscr{B}} \partial(B),$$

therefore

$$\begin{aligned}\bigcup \mathcal{C}'(\mathscr{B}, n-1) &= \partial\left(\bigcup \mathscr{B}\right) \cap \bigcup_{B \in \mathscr{B}} \partial(B) \\ &= \partial\left(\bigcup \mathscr{B}\right).\end{aligned}$$

□

Lemma 3. *Given a set $\mathscr{B}$ of n-dimensional closed balls with pairwise distinct centers, the set $P = \{\mathcal{C}(\mathscr{B}, 0), \ldots \mathcal{C}(\mathscr{B}, n-1)\}$ is a partition of $\partial(\bigcup \mathscr{B})$. In other words, $\bigcup P = \partial\left(\bigcup \mathscr{B}\right)$ and $\bigcap P = \{\}$ if $|P| > 1$.*

Proof. This proof can be made by induction on the dimension.

The base case is when the dimension $n = 1$, in this case

$$P = \{\mathcal{C}'(\mathscr{B}, 0)\} = \{\mathcal{C}(\mathscr{B}, 0)\},$$

for the reason that $|P| = 1$, we only need to check if $\bigcup P = \partial\left(\bigcup \mathscr{B}\right)$ and this follows directly from Lemma 2.

By definition, for z such that $0 < z < n$ we know that $\mathcal{C}(\mathscr{B}, z) = \mathcal{C}'(\mathscr{B}, z) \backslash \mathcal{C}'(\mathscr{B}, z-1)$, and for this reason, $\mathcal{C}(\mathscr{B}, z-1) \cap \mathcal{C}(\mathscr{B}, z)$ is empty because $\mathcal{C}(\mathscr{B}, z-1) \subseteq \mathcal{C}'(\mathscr{B}, z-1)$. Therefore $\bigcap P = \{\}$ for $|P| > 1$.

The union of set P is

$$\begin{aligned}\bigcup P &= \bigcup_{z=0}^{n-1} \mathcal{C}(\mathscr{B}, z) \\ &= \mathcal{C}'(\mathscr{B}, 0) \cup (\mathcal{C}'(\mathscr{B}, 1) \setminus \mathcal{C}'(\mathscr{B}, 0)) \cup \cdots \cup (\mathcal{C}'(\mathscr{B}, z) \setminus \mathcal{C}'(\mathscr{B}, n-1)) \\ &= \mathcal{C}'(\mathscr{B}, 0) \cup \mathcal{C}'(\mathscr{B}, 1) \cup \cdots \cup \mathcal{C}'(\mathscr{B}, n-1) \\ &= \mathcal{C}'(\mathscr{B}, n-1),\end{aligned}$$

and by Lemma 2, we have that $\bigcup P = \partial(\bigcup \mathscr{B})$. □

Lemma 4. *Given the set $\mathscr{J} \subseteq \mathscr{B}$, where $\mathscr{B}$ is a set of n-dimensional closed balls with pairwise distinct centers, the set $\mathcal{S}(\mathscr{J})$ is n-dimensional sphere if and only if $|\mathscr{J}| = 1$.*

Proof. The only way that to $\mathcal{S}(\mathscr{J})$ be a n-dimensional sphere is that all balls in $\mathscr{J}$ are the same, but this is only possible if $|\mathcal{S}(\mathscr{J})| = 1$ because all balls in $\mathscr{J}$ have distinct centers.

If $\mathscr{J} = \{B\}$, then $\mathcal{S}(\mathscr{J}) = \partial(B)$, therefore a n-dimensional sphere. □

To simplify the following definition, we will denote the set of the closest points in the set S to a point p as

$$\textsc{cp}(S, p) = \{q : \forall q' \in S, d(p, q) \leq d(p, q')\}. \tag{B.5}$$

Note that $\textsc{cp}(S, p)$ can be a piece of a n-dimension hypersphere, a point, or empty. Also that if $S \subseteq S'$ where S' is a sphere centered at p, then $\textsc{cp}(S, p) = S$. Additionally, if $|\textsc{cp}(S, p)| > 1$ than S is contains a subset of a sphere centered at p with radius $d(p, S)$.

Finally, given a ball $B \in \mathscr{B}$ and a point $p \in B$ we will denote the set of the closest points from p that are in the boundary of B and in the boundary of $\bigcup \mathscr{B}$ as

$$\textsc{ci}(\mathscr{B}, B, p) = \textsc{cp}\left(\partial(B) \cap \partial\left(\bigcup \mathscr{B}\right), p\right). \tag{B.6}$$

We extend this notation to $\textsc{ci}(\mathscr{B}, B, p) = \{\}$ if $p \notin B$ or $B \notin \mathscr{B}$.

Lemma 5. *Given the set of closed n-dimensional balls $\mathscr{B}$ and a point $p \in \bigcup \mathscr{B}$, if exists a ball $B \in \mathscr{B}$ where $\textsc{ci}(\mathscr{B}, B, p) \neq \{\}$ then a ball S centered at p with radius $d(p, \textsc{ci}(\mathscr{B}, B, p))$ is completely inside $\bigcup \mathscr{B}$.*

Proof. This follows directly from construction of Equation (B.5), if $\textsc{ci}(\mathscr{B}, B, p)$ is non-empty then $\partial(B) \cap \partial(\bigcup \mathscr{B})$ cannot be empty. Then every point in $\textsc{ci}(\mathscr{B}, B, p)$ is in the boundary of B and for the reason that $B \in \mathscr{B}$ then a ball with radius $d(p, \textsc{ci}(\mathscr{B}, B, p))$ is completely inside $\bigcup \mathscr{B}$. □

Lemma 6. *Given the set of closed n-dimensional balls $\mathscr{B}$ and a point $p \in \bigcup \mathscr{B}$, if exists two balls $B, B' \in \mathscr{B}$ such that $\textsc{ci}(\mathscr{B}, B, p) \neq \{\}$ and $\textsc{ci}(\mathscr{B}, B', p) \neq \{\}$, then $d(p, \textsc{ci}(\mathscr{B}, B, p)) = d(p, \textsc{ci}(\mathscr{B}, B', p))$.*

Proof. As a result of Equation (B.5), every point $q \in \text{CI}(\mathscr{B}, B, p)$ has the same distance to p, and similarly, every point $q' \in \text{CI}(\mathscr{B}, B', p)$ has the same distance to p. Given a ball S centered at p with radius equal to w, where $w = d(p, \text{CI}(\mathscr{B}, B, p))$, we can safely say that $S \subseteq B$ due to Lemma 5. The same applies to the ball S' centered at p with radius $w' = d(p, \text{CI}(\mathscr{B}, B', p)$.

Without loss of generality, if $w > w'$ then any point $q' \in \text{CI}(\mathscr{B}, B', p)$ has distance w' from p and w' is smaller than w. Hence, $q' \in \text{int}(S)$ which contracts the construction in equation *Equation* (B.6), because q' is not in $\partial(\bigcup \mathscr{B})$ by the fact that $\text{int}(S) \cap \partial(\bigcup \mathscr{B}) = \{\}$. If $w < w'$ the same argument applies.

Therefore, w must be equal to w'. □

We are only interested in the distance and if CI is empty for all balls in $\mathscr{B}$. Therefore, accordingly with Lemma 6 we will denote by dropping the second argument of CI the set

$$\text{CI}(\mathscr{B}, p) = \text{CP}\left(\partial\left(\bigcup_{\substack{B \in \mathscr{B} \\ p \in B}} B\right) \cap \partial\left(\bigcup \mathscr{B}\right), p\right) \tag{B.7}$$

Lemma 7. *Given the set of closed n-dimensional balls $\mathscr{B}$ and a point p, if $\text{CI}(\mathscr{B}, p) = X$ and $X \neq \{\}$ then the distance from p to the boundary of $\bigcup \mathscr{B}$ is $d(p, X)$.*

Proof. As a result of Lemma 5 a closed ball S with radius $d(p, \text{CI}(\mathscr{B}, p))$ centered in p is completely inside $\bigcup \mathscr{B}$, hence the distance $d(p, \partial(\bigcup \mathscr{B})) \geq d(p, \text{CI}(p, \mathscr{B}))$.

By construction of the *Equation* (B.6), $\text{CI}(p, \mathscr{B})$ has only points that are in $\partial(\bigcup \mathscr{B})$, it is not possible that $d(p, \partial(\bigcup(\mathscr{B}) > d(p, \text{CI}(p, \mathscr{B}))$ because $\text{CI}(p, \mathscr{B}) \subseteq \partial(\bigcup \mathscr{B})$.

Therefore, $d(p, \text{CI}(\mathscr{B}, p))$ is equal to $d(p, \partial(\bigcup \mathscr{B}))$. □

Lemma 8. *Given a non-empty set of closed n-dimensional balls $\mathscr{B}$ and a point p, if $\text{CI}(\mathscr{B}, p) = \{\}$ then $\mathcal{C}'(\mathscr{B}, n-2) \neq \{\}$.*

Proof. This proof is given by showing that exists at least two balls in the set $\mathscr{B}$ where the intersection between them is not empty. If $\text{CI}(\mathscr{B}, p)$ is empty, then we know that exists a non-empty set of balls that intersects every ball that contains p. Therefore, $\mathcal{C}'(\mathscr{B}, n-2)$ is not empty. □

Lemma 9. *Given a non-empty set of closed n-dimensional balls $\mathscr{B}$ and a point $p \in \bigcup \mathscr{B}$, if $\text{CI}(\mathscr{B}, p) = \{\}$ then $\mathcal{C}(\mathscr{B}, n-1) \nsubseteq \text{CP}(\partial(\bigcup \mathscr{B}), p)$.*

Proof. There is two possible alternatives:

(a) Exists a ball $B \in \mathscr{B}$ where $p \in B$ and $\partial(B) \cap \partial(\bigcup \mathscr{B}) \neq \{\}$;

(b) Does not exist such a ball.

For the case (a), we know that $\text{CP}(\partial(B), p)$ is a point because if p is the center of B and $\partial(B) \cap \partial(\bigcup \mathscr{B}) \neq \{\}$ then $\text{CI}(\mathscr{B}, p)$ is also not empty.

The closest point $\text{CP}(\partial(B), p)$ is not in the boundary of $\bigcup \mathscr{B}$ but the there is points in the boundary of B in the boundary of $\bigcup \mathscr{B}$. Therefore exists a set of balls $\mathscr{I} \subset \mathscr{B}$ that intersects B and $\text{CP}(\partial(B), p) \in \bigcup \mathscr{I}$, otherwise $\text{CI}(\mathscr{B}, p)$ could not be empty.

The closest point to p in the boundary of $\bigcup \mathscr{B}$ is the closest point in the boundary of B closest to $\text{CP}(\partial(B), p)$. This point is in the intersection of B and $\bigcup \mathscr{I}$, therefore for this case, the closest point is not in the $\mathcal{C}(\mathscr{B}, n-1)$.

For the case (b), we know that $p \notin B$ for every ball $B \in \mathscr{B}$ such that $\partial(B) \cap \text{CP}(\partial(\bigcup \mathscr{B}, p) \neq \{\}$. For every B, the object $\text{CP}(\partial(B) \cap \partial(\bigcup \mathscr{B}, p)$ is a piece of n-dimensional sphere that is not a complete sphere S. For the reason that $p \notin B$ then, the closest point on the correspondent S must be a point of intersection, otherwise $\text{CI}(\mathscr{B}, p)$ is not empty.

□

Theorem 2. *Given the set of n-dimensional closed balls $\mathscr{B}$ and a point $p \in \bigcup \mathscr{B}$, the distance from p to the boundary of $\bigcup \mathscr{B}$ is*

$$d(p, \partial(\mathscr{B})) = \begin{cases} d(p, \text{CI}(\mathscr{B}, p)) & \text{if } \text{CI}(\mathscr{B}, p) \neq \{\}, \\ d(p, \mathcal{C}'(\mathscr{B}, n-2)) & \text{if } \text{CI}(\mathscr{B}, p) = \{\} \end{cases} \tag{B.8}$$

Proof. This proof follows directly from Lemmas 7 – 9. By Lemma 7, if $\text{CI}(\mathscr{B}, p) \neq \{\}$ the distance is $d(p, \text{CI}(\mathscr{B}, p))$. Otherwise, we have the guarantee that $\mathcal{C}'(\mathscr{B}, n-2)$ is not empty by *Lemma* 8 and that we do not need, for this case, to check candidates of dimension n by Lemma 9. □

B.3 Algorithm

In this section, we will explore the construction of an algorithm to solve the largest ball problem. As a result of Lemma 1, we can reduce the problem to compute the distance from a point p to the boundary of the union of balls, that is $\partial(\bigcup \mathscr{B})$.

The ideia is to implement the Equation (B.8) of Theorem 2, but in practice, the term $\mathcal{C}'(\mathscr{B}, n-2)$ is a complex object. Therefore, we begin with candidates with high dimension, and decrease the dimension if needed.

Algorithm 17 implements the behaviour of the function of Equation (B.8). Lines 1 – 5 implements the first case of the equation, that is, when $\text{CI}(\mathscr{B}, p) \neq \{\}$) and, correspondingly, Lines 6 – 20 implements when $\text{CI}(\mathscr{B}, p) = \{\}$.

The function `FIND-CLOSEST-POINT` in Lines 3 and 11 is a function that will find the closest point from p to an object S, that is, will find a point q where $q \in \text{CP}(S, p')$.

In the case where p' is not the center of S, the function can just compute the point

$$q = \frac{p' - c}{\|p' - c\|} r$$

where c is the center of S and r is the radius.

If p' is the center of S, then $|\text{CP}(S, p')| > 1$. In such situation, the function can just return some point in the boundary of $\partial(S)$. In this case, it is possible that $S \cap \partial(\bigcup \mathscr{B}) \neq \{\}$ and $q \notin \partial(\bigcup \mathscr{B})$, but we have the guarantee that a point will be found in lower dimensions because of Lemma 9.

Lines 6 – 20 describes the procedure that will test the distance to candidates of dimension $n - i + 1$. Note that the first iteration begins with $i = 2$, we can safely do that because of the results of Theorem 2.

In the worst case, in Lines 6 – 20 candidates of every dimension will be tested, but in other cases we can stop the algorithm. Line 20 shows when the algorithm may break.

The variable next-dimension controls whether the loop will continue to other candidates of lower dimension. The min-candidate variable holds the value of the candidate of the current dimension that is closer but has an intersection. Essentially, the loop stops when the closest candidate is in the boundary of $\bigcup \mathscr{B}$.

Algorithm 17: DISTANCE-TO-BOUNDARY

Input: A set of n-dimensional balls $\mathscr{B}$ and a point p such that $p \in \bigcup \mathscr{B}$
Output: The distance of p to the $\partial(\bigcup \mathscr{B})$.

```
1  for each B in 𝓑 do
2      if p ∈ B then
3          q = FIND-CLOSEST-POINT(∂(B), p)
4          if q ∉ int(⋃𝓑) then
5              return d(p, q)
6  min = ∞
7  for each i in {2, ..., n} do
8      next-dimension = false
9      min-candidate = ∞
10     for each S in {⋂∂(J) : J ⊆ 𝓑 ∧ |J| = i} \ {} do
11         q = FIND-CLOSEST-POINT(S, p)
12         dist = d(p, q)
13         if dist < min then
14             if q ∈ int(⋃𝓑) then
15                 next-dimension = true
16                 min-candidate = dist
17             else
18                 min = dist
19                 if min < min-candidate then next-dimension = false
20     if next-dimension = false then break
21 return min
```

Given that $|\mathscr{B}| = k$, the Lines 1 – 5 with the direct implementation is in order of $O(k^2)$ in time. The number of loops in the worst case is the size of the set of intersections constructed in Line 10 is $O(\binom{k}{i})$, therefore, Lines 6 – 20 is

$$
\begin{aligned}
&O\left(\sum_{i=1}^{n} \binom{k}{i} k\right) \\
=&O\left(\binom{k}{n} k\right).
\end{aligned}
$$